De la mobilité à la sécurité des réseaux

Claude Castelluccia

MEMOIRE
De la Mobilité à la Sécurité des Réseaux

présenté dans le cadre de l'école doctorale MSTII pour l'obtention
de l'Habilitation à Diriger des Recherches
de l'Institut Polytechnique de Grenoble (INPG)

Claude Castelluccia

présenté et soutenu publiquement le 10 Septembre 2008

devant le jury composé de

Andrzej Duda . . . . . Président
Luigi Mancini . . . . . Rapporteur
Refik Molva . . . . . Rapporteur
Stephane Ubeda . . . . . Rapporteur

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Structure du document

L'écriture d'une habilitation à diriger des recherches est l'occasion de faire le bilan des activités réalisées et des résultats de recherches obtenus depuis la thèse. Il permet également de prendre du recul afin d'apprécier l'évolution des thématiques abordées et l'impact des résultats. C'est probablement le principal intêret, voire le seul, de cet exercice.

Étant donné qu'il est très difficile de résumer en quelques dizaines de pages les résultats de 12 ans de recherches, j'ai décidé de concentrer ce rapport sur mes activités les plus récentes, c'est à dire sur la sécurité des systèmes sans fil embarqués.

La suite de ce chapitre présente l'évolution de mes travaux de recherches depuis l'obtention de mon doctorat en 1996. Les deux chapitres suivants présentent, plus en détail, mes activités dans le domaine de la sécurité des systèmes embarqués et plus particulièrement des réseaux de capteurs et systèmes RFID. Le dernier chapitre conclut et présente quelques perspectives de recherche. Les annexes contiennent les publications que je considère les plus importantes et dont je suis le plus fier.

1.2 Déjà 12 ans... !

De la mobilité IPv6 à la sécurité IPv6...

Mes premiers travaux à l'INRIA Grenoble ont porté sur les protocoles de gestion de la mobilité dans l'Internet. Un séjour de quelques mois au sein de l'équipe MosquitoNet de l'université de Stanford m'a permis de travailler sur la gestion des interfaces multiples sur les mobiles. J'y ai développé, en collaboration avec professeur Mary Baker et Xinhua Zhao, une solution efficace, basée sur le protocole Mobile IP, qui permet à un terminal mobile de se connecter simultanément à plusieurs réseaux sans fil de différent types (GSM, 802.11, Bluetooth,...). Un mobile peut alors augmenter son débit ou choisir le réseau le mieux adapté à ses besoins [46, 47]. De retour à Grenoble, je me suis intéressé à la gestion de la mobilité dans l'Internet de demain (IPv6) et développé une solution hiérarchique. Ce protocole a ensuite été standardisé à l'IETF, en collaboration avec des chercheurs d'Ericsson Labs, sous le nom de HMIPv6 (Hierarchical Mobile IPv6) et a donné lieu au RFC4140 [43]. J'ai également développé, avec Pars Mutaf (mon troisième thésard), des protocoles de pagination pour l'Internet qui ont conduit à plusieurs publications et drafts Internet [42, 41, 36]. Je me suis parallèlement interessé à la gestion de la mobilité des routeurs embarqués. Ces travaux, en collaboration avec Thierry Ernst (mon premier thésard), ont donné naissance au groupe de travail NEMO de l'IETF (présidé par Thierry Ernst) et à plusieurs drafts Internet [31, 32, 30]. L'arrivée de Imad Aad (mon deuxième thésard) fut l'occasion de mettre en place une activité sur les réseaux locaux sans fil 802.11 et d'étudier le problème de la qualité de service dans ces réseaux. Nous avons alors proposés plusieurs extensions de qualité de service pour les réseaux

802.11 qui ont été publié, entre autre, à Infocom [4, 5, 2, 1].

L'installation de SunLabs Europe en face de l'INRIA à Montbonnot et la rencontre avec Gabriel Montenegro constitua un tournant important dans l'évolution de mes travaux. C'est à ce moment que j'ai commencé à m'intéresser très sérieusement aux problèmes de sécurité dans les réseaux. Ce thème, qui me passionne, est aujourd'hui au coeur de mes travaux de recherche.

Mes premiers résultats dans le domaine de la sécurité fut le développement, en collaboration avec Gabriel Montenegro, des adresses IPv6 cryptographiques, connues sous le nom de SUCV (Statistically Unique and Cryptographically Verifiable addresses) [38, 10, 37]. Ces adresses ont la propriété d'être vérifiables : tout propriétaire d'une adresse IPv6 cryptographique peut effectivement prouver qu'elle lui appartient. Cette vérification est peu coûteuse en ressource et ne repose sur aucun service externe, telle qu'une infrastructure à clefs publiques. Cette solution passe donc très bien à l'échelle et supporte les clients mobiles. Elles peuvent, entre autre, être utilisées pour sécuriser les protocoles de gestion de la mobilité tel Mobile IPv6 [14], sécuriser les communications de groupe [17], éviter le vol d'adresse (IPv6 spoofing) ou simplifier la mise en place de tunnels de chiffrement opportuniste [18]. Les adresses SUCV ont fortement inspiré et ont été à l'origine des adresses CGA (Cryptographically Generated Addresses) qui ont été standardisé à l'IETF.

De la sécurité IPv6 à la sécurité des réseaux Adhoc Mobiles (MANET)....

Mon aventure dans le domaine de la sécurité s'est ensuite concentré sur la sécurité des réseaux mobiles adhoc (MANET) tout d'abord en collaboration avec Gabriel Montenegro, puis en collaboration avec Professeur Gene Tsudik dans le cadre de ma visite à l'université de Californie, Irvine de juillet 2003 à juillet 2005. Nous avons montré avec Gabriel que les adresses SUCV pouvaient être très utiles pour sécuriser les protocoles de routage dans les réseaux MANET [16]. Un réseau MANET est un réseau constitué de machines mobiles qui n'est pas forcément connecté à l'Internet. Chaque noeud est à la fois un terminal et un routeur, et relaie les paquets pour les autres membres du réseau. Étant donné qu'un MANET n'est pas connecté à l'Internet, il est très difficile de les sécuriser. Un des problèmes les plus difficiles est celui de l'admission d'un nouveau membre. Comment autoriser un nouveau noeud à rejoindre le réseau si on ne peut pas utiliser un serveur d'autorisation ? Nous avons alors développé avec Gene Tsudik et ses étudiants un protocole d'admission complètement distribué. Dans ce protocole, un noeud est autorisé à rejoindre un réseau si au moins k des n noeuds du réseau (ou k est un paramètre configurable) donnent un avis favorable sous la forme d'un jeton. En combinant ces k jetons, avec l'aide d'un protocole cryptographique, ce noeud pourra générer son certificat, qui sera vérifiable par les autres membres et lui permettra d'établir des clefs avec chacun d'entre eux. Cette solution, complètement distribuée et autonome, est très peu coûteuse en terme d'énergie et de calcul. Elle est également robuste aux attaques de type déni de service [29]. Ces résultats ont ensuite été complémentés par des travaux sur la protection de la vie privée dans les réseaux MANET. Nous avons développé avec Imad Aad, maintenant chercheur chez NTTDocomo, des algorithmes de routage et de codage qui permettent d'éviter que les communications soient traçables sur un réseau MANET par un attaquant qui a accès à plusieurs noeuds du réseau ou qui peut, tout simplement, écouter les messages échangés entre les noeuds. Avec nos solutions, non seulement un attaquant ne peut pas écouter les messages échangés, car ils sont chiffrés, mais il ne peut pas identifier les noeuds qui communiquent [3].

De la sécurité des réseaux adhoc mobiles à la sécurité des réseaux de capteurs...

Mes travaux sur les réseaux MANETs se sont ensuite peu à peu réorientés sur la sécurité des réseaux de capteurs. Bien que les réseaux MANET et de capteurs soient tous les deux des réseaux ad-hoc multi-sauts et peuvent paraître similaires, ils ont en fait des caractéristiques très différentes. Un réseau de capteurs est typiquement plus grand qu'un réseau MANET : il peut contenir plusieurs centaines, voire plusieurs milliers de noeuds. De plus, un réseau de capteurs est généralement sous le contrôle d'une seule entité. Par conséquence, les problèmes de type "établissement de confiance" sont généralement moins importants. Par ailleurs, les modèles de communication sont différents

dans ces deux types de réseaux. Dans les réseaux MANET, chaque noeud peut potentiellement communiquer avec tous les autres noeuds du réseau. Dans les réseaux de capteurs, un capteur communique géneralement avec une station de base, appelée communément puits, vers lequel il envoie ses mesures. Les communications noeud-à-noeud sont moins fréquentes. De même dans un réseau de capteurs, les noeuds sont rarement mobiles, alors que la mobilité est une caractéristique intrasèque des réseaux MANET. Finalement, les réseaux de capteurs sont caractérisés par des ressources de calcul et surtout d'énergie très limitées. Le défi scientifique le plus important dans les réseaux de capteurs est de développer des protocoles (entre autre de sécurité) qui optimisent ses ressources. Les algorithmes de cryptographie à clef publique y sont souvent proscrits alors qu'ils sont communément utilisés dans les MANETs. Pour résumer, dans les réseaux MANET la problématique vient du fait que les noeuds ne se connaissent pas à priori, et donc ne se font pas confiance, et que le réseau n'est pas forcément connecter à l'Internet. L'utilisation de services tiers, tel qu'une infrastructure à clef publique, n'est donc pas toujours possible. Dans les réseaux de capteurs, les noeuds se font à priori confiance, mais ils ont des ressources très limitées et peuvent être facilement compromis.

Nos travaux sur la sécurisation des réseaux de capteurs ont porté sur différents aspects. J'ai tout d'abord travaillé avec mes collègues de l'Université de Californie sur l'agrégation des données chiffrées. Ce travail a donné naissance au premier algorithme d'agrégation sécurisée de données chiffrées pour réseaux de capteurs [21, 24]. J'ai ensuite développé ce concept avec Aldar Cran (mon premier post-doctorant) et Claudio Soriente, doctorant à l'université de Californie, dans le cadre du projet Européen UbiSec&Sens [26, 11, 45, 27, 6]. J'ai également étudié, avec Pars Mutaf (mon troisième thésard), Aurélien Francillon (mon quatrième thésard) et Angelo Spognardi (mon deuxième post-doctorant), les problèmes d'échange de clefs, de génération de nombres aléatoires [9, 34, 20]. Finallement, nous avons récemment travaillé sur la sécurité des capteurs et avons développé le premier virus/vers pour ce type d'environnement [35]. Ces travaux et résultats sont décrits plus en détail dans le chapitre 2 de ce document.

De la sécurité des réseaux de capteurs à la sécurité des systèmes RFID

Des capteurs aux étiquettes radio-fréquence RFID, il n'y a qu'un petit pas à franchir. Une étiquette RFID permet d'identifier à distance des objets, des animaux ou des personnes. Doté d'une puce contenant jusqu'à 512 bits de mémoire et d'une antenne réagissant aux ondes radio, une étiquette RFID permet en effet de communiquer à distance les informations qu'elle contient, généralement un identifiant.

Bien que les problèmes de recherche liés à la sécurisation des systèmes RFID peuvent, dans un premier temps, sembler similaires à ceux des réseaux de capteurs, ils sont en fait très différents. Ces différences viennent essentiellement du fait, que contrairement aux capteurs, les étiquettes sont mobiles et inertes (c'est à dire ne possède aucune source interne d'énergie). De plus les communications dans les systèmes RFID sont point-à-point, c'est à dire entre l'étiquette et le lecteur, et sont par nature asymétrique (l'étiquette a des capacités très limitées, mais le lecteur est un vrai ordinateur). Finalement, les problèmatiques de sécurité sont différentes. Alors que l'authenticité, l'intégrité et la confidentialité sont les services qui suscitent le plus d'intêret dans les réseaux de capteurs, la protection de la vie privée est la problématique la plus importante des systèmes RFID. En effet, une étiquette RFID renvoit toujours la même valeur aux lecteurs qui l'interrogent : la valeur de son identifiant. Pour des raisons de passage à l'échelle et de performance, cet identifiant est rarement chiffré et la légitimité du lecteur n'est jamais vérifiée. En disposant des lecteurs à plusieurs endroits, le suivi de l'étiquette, et par conséquent de son porteur, devient une réalité. Ce risque a poussé des consommateurs à boycotter les produits Benetton et Wallmart qui souhaitaient équiper certains de leurs produits d'étiquettes RFID.

Mes travaux se sont essentiellement portés sur le problème d'identification secrète des étiquettes très bon marché. Notre objectif était de concevoir des mécanismes et protocoles qui permettent de rendre la traçabilité de ces étiquettes plus difficile, tout en utilisant des étiquettes EPC standards.

J'ai proposé deux solutions : les *étiquettes brouilleuses* (noisy tags) et une solution statistique, appelé *ProbId*. La première solution, développée avec Gildas Avoine de l'EPFL, permet une identi-

fication secrète des étiquettes en environnements surveillés (banques, entreprises, aeroports,...) et protège contre des attaquants passifs [13]. La deuxième solution, développé avec Mate Soos (mon cinquième thésard), est plus générique et protège les étiquettes également dans tous les environnements ouverts [23, 44]. Elle est cependant un peu plus onéreuse car, bien que ne nécessitant aucun calcul, elle utilise une zone mémoire ROM de quelques centaines de bits.

Ces deux solutions sont présentées en détails dans le chapitre 3.

Et un peu de cryptographie...

Mon séjour à l'Université de Californie en 2003 et 2004 a aussi été l'occasion de faire une escapade dans le domaine de la cryptographie en travaillant avec les professeurs Jarecki et Tsudik sur des améliorations du protocole SSL [22], de nouveaux algorithmes d'agrégation d'acquittements [15] et de "secret handshake" (accord secret) [7, 8]. Nous avons développé de nouvelles constructions cryptographiques qui permettent à deux entités de prouver qu'ils appartiennent à la même organisation sans la dévoiler. Ce type de construction peut, par exemple, être utile à des agents des services secrets qui souhaiteraient s'identifier sans révéler leur affiliation à des membres extérieurs à leur organisation. Les protocoles qui utilisent des certificats standards sont, bien entendu, mal adaptés, car la vérification des certificats n'est pas secrète et peut être effectué par tout le monde. Avec notre solution, si le protocole se termine correctement alors les deux entités ont la certitude qu'ils appartiennent à la même organisation. Une entité malicieuse qui écouterait l'échange n'obtiendrait aucune information. Si le protocole échoue, les entités peuvent en conclure qu'il n'appartiennent pas à la même organisation et ne sont pas en mesure d'identifier leur organisation d'appartenance respective.

Nous avons également développé des algorithmes d'agrégation d'acquittement pour les communications de groupe. Lorsqu'une source émet une donnée à un groupe et souhaite s'assurer qu'elle est bien reçue par tous les membres, chaque récepteur doit répondre avec un acquittement. Ces acquittements peuvent créer le problème bien connu d'implosion à la source, qui doit alors traiter un nombre important de messages. Pour résoudre ce problème, il est souvent recommandé d'agréger les acquittements dans le réseau. Cependant, si les acquittements sont agrégés la source ne peut plus vérifier cryptographiquement l'identité des noeuds qui ont reçus la donnée, car les codes d'authentification ne sont pas agrégeables [27]. En d'autres termes, un noeud intermédaire peut tromper la source en prétendant que certains membres, qui n'ont pas reçu la donnée, l'ont acquittée. Nous avons alors développé un protocole qui permet de résoudre ce problème de façon efficace [15].

Bien que la cryptographie ne soit clairement pas au coeur de mes activités de recherche, cette escapade fut très enrichissante et très instructive. Elle m'a permis de constater les différences de language et d'esprit qui existent entre les chercheurs du domaine des réseaux et ceux du domaine de la cryptographie. Cet expérience fut très utile car un chercheur qui travaille dans le domaine de la sécurité informatique doit, sans forcément contribuer à ce domaine, absolument comprendre la cryptographie. Faute de quoi, il peut faire de mauvaises d'hypothèses et réaliser des erreurs grossières.

Chapitre 2

Sécurité des réseaux de capteurs

2.1 Introduction

Les progrès réalisés ces dernières décennies dans les domaines de la microélectronique, de la micromécanique, et des technologies de communication sans fil, ont permis de produire à un coût raisonnable des composants de quelques millimètres cubes de volume. De ce fait, un nouveau domaine de recherche s'est créé afin d'offrir des solutions économiquement intéressantes et facilement déployables pour la surveillance à distance et le traitement des données dans les environnements complexes et distribués : *les réseaux de capteurs sans fil*. Les réseaux de capteurs sans fil sont constitués de noeuds déployés en grand nombre en vue de collecter et transmettre des données environnementales vers un ou plusieurs points de collecte d'une manière autonome.

Ces réseaux ont un intérêt particulier pour les applications militaires, environnementales, domotiques, médicales, et bien sûr les applications liées à la surveillance des infrastructures critiques. Ces applications ont souvent besoin d'un niveau de sécurité élevé. Or, de part de leurs caractéristiques (absence d'infrastructure, contrainte d'énergie, topologie dynamique, nombre important de capteurs, sécurité physique limitée, capacité réduite des noeuds,...), la sécurisation des réseaux de capteurs est à la source, aujourd'hui, de beaucoup de défis scientifiques et techniques.

2.2 Réseaux de Capteurs

Un réseau de capteurs est composé de centaines ou de milliers d'ordinateurs minuscules. Ces appareils, appelés en anglais **motes**, sont alimentés par des piles et sont typiquement déployés de façon aléatoire dans des environnements souvent ouverts. Ces capteurs font généralement des mesures périodiques et envoient les données collectées à un dispositif plus puissant, le puits, qui les traite en calculant par exemple leur maximum, moyenne ou médiane.

Les noeuds qui composent un réseau de capteurs sans fil sont petits et par conséquent ont des ressources de calcul, de stockage, de communication et d'énergie très limitées. L'architecture de système d'un noeud de capteur sans fil est composée de quatre éléments : (i) Un sous-système de calcul composé d'un microprocesseur ou d'un microcontrôleur, (ii) un sous-système de communication composé d'une radio de courte portée (iii) un sous-système de mesure qui lie le noeud avec le monde physique, et (iv) un sous-système d'alimentation d'énergie, qui loge la pile et le convertisseur DC-DC, et qui alimente le reste du noeud.

Les applications potentielles des réseaux de capteurs sont les applications militaires et les applications de surveillance de l'environnement (monitoring). Par exemple, ils peuvent être utilisés pour la détection de feux dans des grandes zones forestières, l'observation d'environnements naturels (pollution, inondation, etc.), suivi d'écosystèmes (surveillance d'oiseaux, croissance des plantes, etc.), contrôle militaire (télésurveillance de champs de bataille, détection d'ennemis, etc.), analyses biomédicales et surveillance médicale (détection de cancer, rétine artificielle, taux de glucose, diabètes, etc.). Un des intêrets des réseaux de capteurs est qu'ils peuvent être déployés dans des

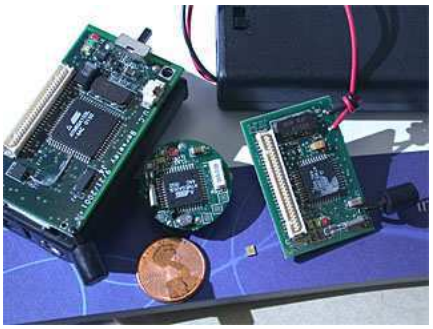


FIG. 2.1 – Berkeley Mote

endroits difficiles d'accès. Ils peuvent, par exemple, être jetés d'un hélicoptère, comme illustré par la Figure 2.2.

Les réseaux de capteurs ont aussi beaucoup d'applications dans le domaine de la santé. Ils peuvent, par exemple, être utilisés pour surveiller à distance des patients. Dans ce cas, ils permettent non seulement d'améliorer la qualité de vie des malades, qui peuvent rester chez eux, mais aussi intervenir le plus rapidement possible si les mesures effectuées par les capteurs sont anormales. A titre d'exemple, Intel travaille sur un projet de recherche dans le but est d'assister les personnes agées. Dans ce projet, chaque objet de la maison (tasses, assiettes, chaises,...) est équipé d'un micro-capteur et enregistre les activités quotidiennes de l'occupant. Les capteurs envoient ensuite les données à un système centralisé qui permet alors à un infirmier de contrôler en permanence et à distance les activités de l'occupant.

Toutes ses applications ont des contraintes de sécurité très différentes. Cependant, dans la plupart d'entre elles, l'intégrité et l'authenticité des données doivent être fournies pour s'assurer que des noeuds non-autorisés ne puissent pas injecter des données dans le réseau. Le chiffrement des données est souvent requis pour des applications sensibles telles que les applications militaires ou les applications médicales.

Dans ce document, nous nous intéressons essentiellement à la sécurisation du réseau (c'est à dire de l'insfrastructure). Il faut cependant noter que sécuriser le réseau est nécessaire mais pas suffisant. En effet, un attaquant peut, dans certains cas, facilement modifier l'environnement et injecter des données erronées. Par exemple, dans le cas d'un réseau utilisé pour la détection de feux dans une forêt, un attaquant peut chauffer plusieurs capteurs avec un briquet et générer un fausse alerte. Par conséquent, le réseau doit aussi utiliser des tests de plausibilité qui permettent de vérifier que les mesures obtenues sont cohérentes. Ces tests sont généralement réalisés par le puits (sink).

La sécurisation des réseaux de capteurs est un problème difficile pour les raisons suivantes :

1. **Capacités limitées** :

   Les ressources de calcul et de mémoire des noeuds sont relativement faibles. Par exemple, les noeuds capteurs de type "mote" sont composés d'un micro-controleur 8-bits 4MHz, 40 KOctets de mémoire et une radio avec un débit d'environ 10 kbps. Même les noeuds de moyenne gamme, comme UCLA/ROCKWELL'S WINS, ont un processeur StrongARM 1100

FIG. 2.2 – Déploiement de capteurs par hélicopter

avec une mémoire flash de 1 MO, une mémoire RAM de 128 KO et une radio de 100 Kbps. Non seulement les capacités des noeuds sont faibles, mais en plus ils sont alimentés par des piles et par conséquent ont une durée de vie limitée. L'énergie limitée des capteurs est probablement la caractéristique la plus pénalisante. Le plus grand des défis dans le domaine des réseaux de capteurs reste de concevoir des protocoles, entre autre de sécurité, qui minimisent l'énergie afin de maximiser la durée de vie du réseau. En d'autres termes, l'énergie est sans aucun doute la ressource qui convient de gérer avec la plus grande attention.

Les solutions de sécurité qui existent aujourd'hui ne sont pas utilisables car elles sont souvent trop coûteuses en terme de ressource. Par exemple, l'utilisation de la cryptographie à clés publiques est souvent proscrite de ce type d'environnement. De nouveaux algorithmes et protocoles de sécurité sont nécessaires.

2. **Agrégation des données** : Il a été montré dans plusieurs publications scientifiques que la transmission d'un bit est équivalent, en terme d'énergie, à l'execution d'environ 1000 instructions. Cette valeur augmente avec la portée de la radio. Plus le capteur devra transmettre loin, et par conséquent augmenter sa puissance d'émission, plus il va consommer de l'énergie, et par conséquent réduire sa durée de vie. Il convient donc de réduire en compressant ou en agrégeant les données lors de leur routage.

   Les techniques d'agrégation des données, c'est à dire de traitement des données par le réseau, permettent de réduire le nombre de messages (et de bits transmis sur les liens sans-fil) et par conséquent réduire la consommation en énergie. Par exemple, si un réseau est déployé pour mesurer la température et que le puits n'est interessé que par la moyenne des températures, un noeud intermédiaire pourra additionner les valeurs reçues de ses enfants et envoyer le résultat à son père. Le puits recevra alors qu'un seul message, contenant la somme des données au lieu de n messages (ou n est le nombre de capteurs).

   Ces techniques d'agrégation sont souvent utilisées. Elles sont cependant difficiles à mettre en oeuvre lorsque les données sont chiffrées car le traitement des données devient alors très délicat.

3. **Echelle et dynamicité** : Les réseaux de capteurs contiennent souvent un nombre de noeuds très important. Ces réseaux sont souvent peu stables et très dynamiques : les capteurs, qui

ont consommé leur pile, disparaissent et de nouveaux noeuds doivent être déployés pour assurer une certaine connectivité.

4. **Protection physique faible** : Les capteurs sont souvent déployés dans des environnements non-protégés (montagnes, forêts, champs de bataille,...). Par conséquent, ils peuvent facilement être interceptés et corrompus. De plus, à cause de leur faible coût, ils utilisent rarement des composants électroniques anti-corruption (tamper-resistant devices).

2.3 Établissement de liens sécurisés

Cette section présente certaines de nos contributions sur l'établissement de liens sécurisés dans les réseaux de capteurs. Nous considérons, plus particulièrement, le problème d'échange de clés.

Cette section est composée de deux parties qui diffèrent par le type de réseaux et capteurs utilisés. La première partie concerne les capteurs dont les clés peuvent être installées avant déploiement. C'est souvent le cas pour les applications domotiques, lorsque, par exemple, un utilisateur souhaite configurer une clé entre un appareil (lampe, prise,...) et sa télé-commande. Le défi, ici, est de proposer une solution qui soit facile d'utilisation pour un utilisateur inexpérimenté, peu coûteux en ressource, et qui ne nécessite pas d'appareillage particulier. Nous présentons notre solution "Shake Them Up!" (STU). La deuxième partie partie traite du problème d'échange de clés lorsque les noeuds sont déployés aléatoirement. Chaque noeud doit établir une clé secrète avec chacun de ces voisins. Ce problème est difficile sachant que l'utilisation de la cryptographie à clé publique n'est pas possible. Nous présentons le protocole "RoK", une solution qui améliore la sécurité de la solution d'Eschanauer et Gligor, qui est aujourd'hui le standard de-facto [33].

2.3.1 Le protocole Orangina : bien secouer avant utilisation ! [20, 25]

Dans beaucoup d'applications de réseaux de capteurs, il est nécessaire de configurer les noeuds avec des clés secrètes avant leur déploiement. Cette phase de "pairing" n'est pas triviale car les noeuds de capteurs n'ont généralement pas d'interface qui permet cette configuration. Les solutions existantes ne sont pas satisfaisantes car elles nécessitent soit d'utiliser du matériel coûteux et mobile (cage de faraday), des interfaces qui n'existent pas sur les capteurs (écran et/ou clavier), l'utilisation de cryptographie à clé publique (trop coûteux pour les capteurs) ou des liens de communication qui sont difficilement exploitables (canal infra-rouge ou contact électrique).

L'objectif de ce travail consistait à développer un nouveau protocole d'échange de clés qui ne possède pas les faiblesses des solutions existantes. Plus spécifiquement, la nouvelle solution devait : (1) être facile d'utilisation, (2) avoir un coût en terme de ressource (mémoire, CPU, batterie) très faible et (3) ne pas utiliser des composants matériels additionnels (cage de faraday, nouvelle interface radio,...).

Nous avons développé un nouveau protocole, appelé *STU* (Shake Them Up!), qui répond à toutes ses contraintes. Ce résultat a fait l'objet d'une publication à la conférence ACM Mobisys. Cette solution, très originale et innovante, eut un certain succès dans notre communauté scientifique. Elle est aujourd'hui utilisée et référencée par de nombreux projets de recherche.

L'idée de base de ce protocole est décrit comme suit : Deux noeuds A et B communiquant sur un canal qui fournit l'*anonymat des sources* peuvent s'échanger un bit secret en utilisant l'algorithme suivant :

- A et B s'échangent le bit secret "1" si A envoie le message 01 ou B envoie le message 10.
- A et B s'échangent le bit secret "0" si A envoie le message 10 ou B envoie le message 01.

Toute autre noeud C qui écoute l'échange ne peut pas identifier le bit échangé. En effet, comme le canal assure l'anonymat des sources, C ne peut pas identifier la source du message. S'il entend le message 01, étant donné qu'il ne sait pas si ce message a été envoyé par A ou B, il ne peut pas savoir si le bit secret est 0 ou 1 ! En revanche, A (resp. B) peut identifier la source du message : s'il n'a pas envoyé le message...il a forcément été envoyé par B.

En généralisant ce protocole à n échanges, A et B peuvent s'échanger une clé secrète de n bits. Figure 2.3 illustre notre protocole. Le protocole est initié par un message *start* émis par un des

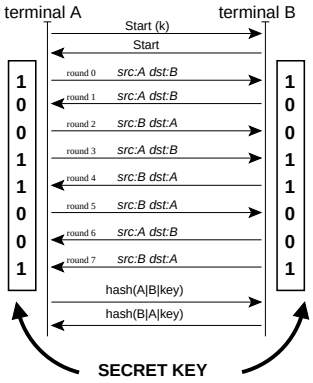


FIG. 2.3 – Protocoles d'échange de clés STU.

participants. Ensuite, le protocole est divisé en k période de temps, ou k est la taille de la clé. A chaque période de temps, le noeud A (resp. B) initie une temporisation. Si la temporisation arrive à écheance avant la réception d'un message de B, il envoie un message contenant la valeur 01 (resp. 10) s'il veut transmettre le bit secret 0 (resp. 1) ou la valeur 10 s'il veut transmettre le bit secret 1 (resp. 0). Au bout des k périodes, A et B partagent une clé secrète key (10011001 dans l'exemple de la figure). Chacun calcule alors la valeur $h(A|B|key)$, ou $h(.)$ est un fonction de hachage comme $SHA1$, et s'échange cette valeur. Ils peuvent alors s'assurer qu'ils ont tous les deux obtenus la même clé.

Le protocole suivant fait l'hypothèse que le canal assure l'anonymat des sources. Cette propriété est possible sur un lien sans-fil sous certaines conditions :

1. L'analyse temporelle, et notamment la couche d'accès au médium (Medium Access Control) ne doit pas fournir d'information sur la source d'un paquet. Cette propriété n'est pas assurée avec les systèmes TDMA (Time Division Multiplexing Access) car, avec ces protocoles, chaque source doit émettre uniquement durant des intervals préalablement attribués. Ces intervals d'émission peuvent donc être utilisés pour identifier la source. Par contre, les protocoles de type CSMA (Collision Sense Multiplexing Access) assurent cette propriété, car les sources accèdent au canal de façon aléatoire. Le protocole STU ne fonctionne donc qu'avec des systèmes utilisant CSMA (Collision Sensing Multiple Access), comme 802.11.
2. L'analyse spatiale ne doit fournir aucune information sur la source. Cette propriété est difficile à assurer en pratique car les ondes radio s'atténuent de façon inversement proportionnelle au carré de la distance. Par conséquence, la puissance de réception d'un message fournit beaucoup d'information sur la localisation d'un noeud. Pour résoudre ce problème, nous proposons qu'un utilisateur qui souhaite établir une clé entre deux noeuds A et B, rapproche et secoue A et B pendant l'exécution du protocole STU décrit précédemment. En secouant A et B, et en faisant l'hypothèse que ces 2 noeuds soient de même type et émettent à la même puissance, les puissances de réception des messages de A et B s'égalisent et ne fournissent aucune information sur la source des messages.

L'article scientifique STU décrit notre protocole en détail [25]. Il présente une analyse théorique de la sécurité et quelques résultats expérimentaux. Il met également en évidence certaines attaques possibles et présente les contre-mesures à utiliser. Finalement, il décrit une extension du protocole STU qui permet l'établissement d'une clé de groupe entre plusieurs noeuds d'un réseau.

2.3.2 RoK : Un protocole d'échange de clés pour réseaux de capteurs [9]

Le travail précédent a traité le problème de l'établissement de clés secrètes entre noeuds avant leur déploiement. Cependant, dans certaines applications, l'établissement des clés doivent se faire une fois que les capteurs sont déployés. C'est le cas lorsque qu'un grand nombre de capteurs est déployé de façon aléatoire, par exemple à partir d'un hélicoptère. Dans ce type de scénario, chaque noeud doit établir une clé secrète avec chacun de ses voisins. Ses voisins n'étant pas connus avant le déploiement, ces échanges de clés doivent avoir lieu sur le terrain. Ce problème serait trivial à résoudre si l'utilisation des protocoles utilisant la cryptographie à clé publique était possible. Les noeuds pourraient alors s'échanger leur certificat, un composant de type Diffie Hellman et calculer une clé secrète. Malheureusement, l'utilisation d'algorithme utilisant la cryptographie à clé publique est trop coûteuse et ne peut être utilisée dans ce type d'environnement.

Pour résoudre ce problème, Eschenauer et Gligor ont proposé, en 2002, un protocole probabiliste qui est devenu le standard de-facto [33]. Ce protocole, qui est relativement simple, opère comme suit :

1. L'administrateur du réseau génère un tableau de l nombres aléatoires, indexés de 1 à l.
2. Chaque noeud est ensuite configuré avec un sous-ensemble m de clés choisies aléatoirement parmi les l clés précédentes.
3. Lorsque deux noeuds, A et B, veulent établir une clé secrète, ils s'échangent les indexes de leurs clés. Ils utilisent ensuite les clés qu'ils ont en commun pour générer un secret (par exemple en les hachant en utilisant un fonction de hachage comme SHA1).

Ce protocole est probabiliste car il est toujours possible qu'un noeud, autre que A et B, ait également été configuré avec les clés que A et B possèdent en commun. Ce noeud pourra alors calculer le secret que A et B viennent d'établir. Cependant Eschanauer et Gligor ont montré que cette probabilité peut être faible si les paramètres l et m sont choisis avec précaution.

Nous avons montré que, bien que ce protocole soit efficace, la sécurité qu'il fournit se dégrade rapidement avec le temps. En effet, chaque fois qu'un noeud est compromis, ses clés sont révélées. En fil du temps, et en faisant l'hypothèse que l'attaquant compromet continuellement des noeuds, une grande partie des clés du système sera connu de l'attaquant. L'attaquant peut, alors, calculer les clès établis entre les noeuds et lire le contenu des messages échangés ou insérer des messages érronés. Dans beaucoup d'applications, de nouveaux noeuds devront être ajouter au fil du temps pour remplacer ceux qui ont consommé leurs piles. La probabilité que les clés de ces nouveaux noeuds soient connues par l'attaquant augmente avec le temps.

Pour résoudre ce problème nous avons proposé *RoK*, une extension du protocole précédent. Cette extension améliore considérablement la sécurité du protocole initiale. Plus spécifiquement, avec *RoK*, le réseau s'auto-guérit lorsque que l'attaque est temporaire (c'est à dire compromet les noeuds uniquement pendant une période de temps, puis disparaît). En d'autre termes, l'état de réseau redevient sain lorsque l'attaquant disparaît. Par ailleurs, l'état du réseau se stabilise lorsque l'attaquant est permanent (c'est à dire compromet des noeuds continuellement sans s'arrêter).

Les figures 2.4 et 2.5 montrent les résultats de quelques simulations (pour plus de détails sur ces simulations, le lecteur pourra lire l'article *RoK* [9], inclut dans ce document). La figure 2.4 montre la proportion de liens compromis en fonction du temps (c'est à dire en fonction du nombre de noeuds compromis) pour un attaquant *permanent*. Les différentes courbes correspondent aux résultats obtenus avec le protocole standard et le protocole RoK, pour différents nombres de compromissions par période de temps. Ces résultats montrent que la sécurité du protocole standard se dégrade avec le temps et qu'au bout d'un certain nombre de périodes de temps (variable en fonction de l'aggressivité de l'attaquants), tous les liens du réseau sont compromis ! En comparaison,

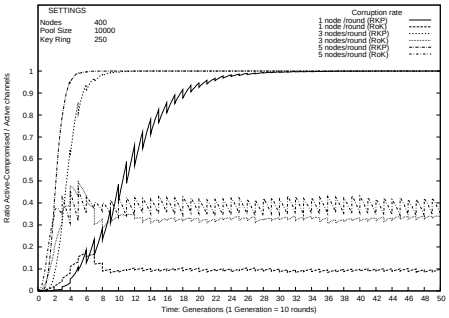


FIG. 2.4 – Ratio de liens compromis avec attaquant permanent

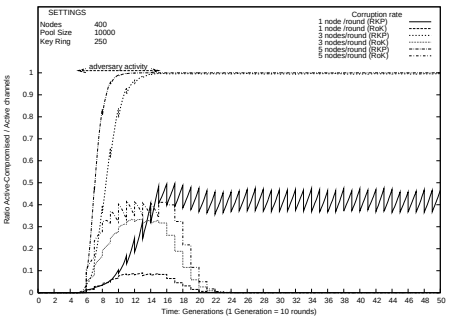


FIG. 2.5 – Ratio de liens compromis avec attaquant temporaire.

avec le protocole RoK, le pourcentage de liens compromis converge vers une valeur comprise entre 10% et 40%.

La figure 2.5 montre la proportion de liens compromis en fonction du temps pour un attaquant *temporaire*. Dans nos simulations, l'attaquant est présent entre les périodes 5 et 15, puis disparaît. Les différentes courbes correspondent aux résultats obtenus avec le protocole standard et le protocole *RoK*, pour différents nombres de compromissions par période de temps. Ces résultats montrent que la sécurite du protocole standard se dégrade avec le temps et qu'au bout d'un certain nombre de périodes de temps (variable en fonction de l'aggressivité de l'attaquants), tous les liens du réseau sont compromis ! En revanche, avec RoK, le pourcentage de liens compromis augmente entre 10% et 40%, mais converge ensuite vers 0 lorsque l'attaquant disparaît. Ces résultats illustrent bien la propriété d'auto-guérisson du protocole RoK.

L'idée principale du protocole *RoK* est de limiter la durée de vie des clés secrètes contenus dans le tableau et de faire évoluer les clés du système en fonction du temps. Ce choix est motivé par l'observation que la durée de vie d'un capteur est limitée et que par conséquent ses clés n'ont pas besoin d'être permanentes.

Une solution naïve serait que le système change les clés de son tableau à chaque période. A une période donnée T, un noeud pourrait échanger des secrets avec ses voisins et ensuite effacer les clefs de configuration de sa mémoire. Par conséquent, si ce noeud est compromis plus tard, ses clefs ne pourront pas être utilisées. En d'autres termes, l'attaquant serait condammé à être très rapide et compromettre les noeuds dès leur déploiement. Cette solution, bien qu'efficace en terme de sécurité, a un gros inconvénient : un noeud déployé à la période $T + 1$, ne peut pas configurer de secret avec les noeuds déployés pendant les périodes précédentes ! Elle a donc peu d'intêret en pratique.

RoK résout ce problème en attribuant à chaque noeud $w * m$ clés, ou w est la durée de vie moyenne d'un capteur. Si un noeud est déployé à la période T, il sera configuré avec m clés

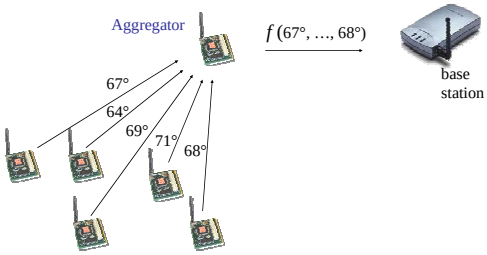


FIG. 2.6 – Agrégation dans réseau de capteurs

de la période T, m clés de la période $T+1$,..., et m clés de la période $T+w$. Ainsi ce noeud pourra échanger un secret avec des noeuds de sa génération, mais aussi des noeuds des génération $T+1$, $T+2$,...,$T+w$. Un des inconvénients de cette approche est qu'elle augmente le nombre de clés que chaque noeud doit stocker par un facteur de w. Pour résoudre ce problème, nous avons développer une nouvelle construction à base de condensés, SHA1 par exemple, doublement chainés. Cette solution permet de réduire le coût mémoire de $w*m$ à $2*m$, c'est à dire un coût constant. L'article [Cast07d] présente le protocole RoK en détail. Il analyse également sa sécurité analytiquement et par simulation.

2.4 Sécurisation des données agrégées

Les réseaux de capteurs sont des réseaux ad-hoc composés de dispositifs minuscules qui ont des capacités de calcul, mémoire et énergie très limitées. Etant donné que la transmission des données est l'opération la plus coûteuse en terme d'énergie, il est essentiel, pour optimiser la durée de vie du réseau, de réduire le nombre de bits envoyés et relayés par les noeuds intermédiaires.

Une approche répandue consiste à agréger les données lors de leur acheminement vers le puits - le noeud qui collecte et traite l'ensemble de données.

Prenons l'exemple d'un réseau qui est déployé pour mesurer la température moyenne dans une zone géographique donnée. Ce réseau est structuré comme un arbre ou les feuilles et les noeuds intermédiaires sont les capteurs et la racine est le puit. Chaque capteur envoie périodiquement ses données vers le puits. Chaque message est relayé noeud par noeud du capteur vers le puits. Par conséquent, si le réseau est constitué de n noeuds, le puits recevra à chaque période de mesure n messages !

Pour réduire le nombre de messages et de bits transmits, chaque noeud intermédiaire peut additionner les données (températures) reçues de ces fils, ajouter la valeur de sa mesure, et envoyer le résultat à son père. Le puits recevra alors un seul message qui contiendra la somme des messages au lieu de n messages (voir Figure 2.6). Il pourra alors diviser cette somme par n et obtenir la moyenne de la température. Si les valeurs mesurées par chaque capteur sont codées sur m bits, le puits recevra $log_2(n)+m$ bits au lieu de $n+m$ bits. Le gain en bande passante est alors de $(log_2(n)+m)/(n+m)$.

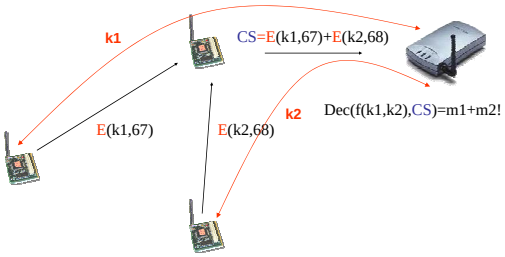


FIG. 2.7 – Agrégation Sécurisée

2.4.1 Agrégation de données chiffrées [21, 24, 26, 11]

L'agrégation des données est relativement triviale, mais devient problématique lorsque l'on veut y ajouter de la sécurité et plus particulièrement de la confidentialité (chiffrement). Dans certaines applications, il est essentiel de s'assurer que les informations qui sont transmises sur le réseau ne puissent être interceptées et lues par des personnes non-autorisées. Elles doivent donc être chiffrées. Mais le chiffrement et l'agrégation sont deux concepts qui ne vont pas très bien ensemble.

Il y a bien entendu deux solutions simples (mais naïves) à ce problème. La première consiste à configurer tous les noeuds du réseau avec une clé de groupe. Chaque noeud chiffre alors ses données avec cette clé en utilisant un algorithme de chiffrement symétrique standard, comme AES ou RC5, et envoie le résultat à son père. Ce noeud déchiffre alors tous les messages reçus de ces enfants, les ajoute, ajoute sa mesure, chiffre le résultat avec la clé de groupe et envoie le message chiffré à son père. Les messages sont ainsi relayés et agrégés de noeud en noeud jusqu'au puits. Le puits peut alors déchiffrer le message et retrouver la somme des données. Cette solution a plusieurs inconvénients : (1) Elle est peu sûre car il suffit de compromettre un seul noeud pour découvrir la clé de groupe et déchiffrer l'ensemble des messages. (2) Elle est peu efficace car chaque noeud doit déchiffrer plusieurs messages et en chiffrer un.

Une deuxième solution consiste à utiliser, au lieu d'une clé de groupe, une clé différente lien par lien. En d'autres termes, dans cette solution, chaque noeud établit une clé secrète avec chacun de ses voisins. Il déchiffre alors les données reçues de ses fils, les agrège, puis chiffre le résultat avec la clé qu'elle possède avec son père. Cette solution est meilleure, en terme de sécurité, que la précédente car la compromission d'un noeud ne révèle que les clés utilisées par ce noeud et non une clé globale. Cependant elle a les inconvénients suivants : (1) La compromission d'un noeud près du puits permet d'obtenir un agrégat qui est significatif car chaque noeud a accès aux données agrégées envoyées par ses fils. (2) Elle nécessite l'établissement de clé entre chaque noeud voisin, ce qui n'est pas trivial. (3) Comme la solution précédente, elle est relativement coûteuse.

Une solution de *bout-en-bout* serait préférable car la compromission d'un noeud ne fournirait aucune information sur l'aggrégat ou les données envoyées par les autres noeuds du système. La solution idéale serait d'avoir une solution ou chaque noeud chiffrerait ses données avec une clé qu'il partagerait avec le puits et avec laquelle les noeuds intermédiaires manipuleraient des données chiffrées sans jamais accéder aux données en clair (voir Figure 2.7). Pour arriver à ce résultat, nous avons besoin d'un fonction de chiffrement homomorphique par l'addition, c'est un

fonction de chiffrement $enc()$ qui posséde la propriété suivante :

$$enc(k_1, m_1) \otimes enc(k_2, m_2) = Enc(f(k_1, k_2), m_1 + m_2)$$

En d'autres termes, la somme des données en clair $m_1 + m_2$, peut être obtenus en déchiffrant avec un clé qui est dépend de k_1 et k_2, le résultat de $\otimes$, une fonction à définir, sur les messages chiffrés m_1 et m_2. Avec une telle fonction de chiffrement, il suffirait alors que chaque noeud exécute la fonction $\otimes$ en utilisant comme paramètres d'entrée, les valeurs chiffrées de ses fils. Le puits pourrait alors obtenir la somme des données en déchiffrant le message qu'il reçoit avec une clé qui dépendrait de l'ensemble des clés qu'il partage avec les noeuds.

Nous avons proposé une fonction de chiffrement qui posséde cette propriété et qui, en plus, est très performante et bien adaptée aux réseaux de capteurs. Nous avons montré qu'en remplaçant dans un chiffrement par flot (stream-cipher) le ou-exclusif *xor* par une addition modulaire (+) nous obtenons une algorithme de chiffrement qui est homomorphique par l'addition et dont le sécurité est prouvable. Cette fonction est décrite dans le tableau 2.4.1.

Additively Homomorphic Encryption Scheme

Encryption :

1. Represent message m as integer $m \in [0, M-1]$ where M is large integer.
2. Let k be a randomly generated keystream, where $k \in [0, M-1]$
3. Compute $c = Enc(m, k, M) = m + k \pmod M$

Decryption :

1. $Dec(c, k, M) = c - k \pmod M$

Addition of Ciphertexts :

1. Let $c_1 = Enc(m_1, k_1, M)$ and $c_2 = Enc(m_2, k_2, M)$
2. For $k = k_1 + k_2$, $Dec(c_1 + c_2, k, M) = m_1 + m_2$

Etant donné que l'addition est une opération commutative, la fonction décrite précédemment est homomorphique par l'addition. En effet, si $c_1 = Enc(m_1, k_1, M)$ et $c_2 = Enc(m_2, k_2, M)$ alors $c_1 + c_2 = Enc(m_1 + m_2, k_1 + k_2, M)$. Si n messages chiffrés sont ajoutés, alors M doit être plus grand que $\sum_{i=1}^{n} m_i$. En effet, si M est plus petit que $\sum_{i=1}^{n} m_i$, le résultat du déchiffrement donnerait une valeur m' inférieure à M et plusieurs valeurs de $m_1 + m_2$ seraient alors possibles. En pratique, si $p = max(m_i)$ alors M doit être égale à $M = 2^{\lceil log_2(p*n) \rceil}$.

L'article scientifique [24] présente notre solution en détails. Il présente également des résultats de performance et la preuve de sécurité de cette nouvelle construction. L'agrégation de données chiffrées est considérée comme un problème scientifique très difficile. Non seulement, nous avons montré qu'il existait une solution, mais nous avons également développé une solution efficace, sûre et peu coûteuse en calcul et mémoire. Cette solution utilise uniquement des additions modulaires et est, par conséquent, très bien adaptée aux réseaux de capteurs. Elle constitue, aujourd'hui, la seule solution pratique à ce problème.

Un inconvénient de cette solution est que le puits doit connaître, pour calculer correctement la clé de déchiffrement, les identités de tous les noeuds qui ont participé à l'agrégat. En d'autres termes, les capteurs qui ont participé doivent envoyer leurs identités. La transmission de ces identifiants peut avoir un coût non négligeable lorsque le réseau est peu stable et que le nombre de capteurs qui participent est très variable. Nous avons développé une optimisation qui permet de réduire le nombre de bits utilisés par l'émission des identifiants [11]. Cette optimisation peut réduire le coût de transmission des identifiants par un facteur pouvant aller jusqu'à 15.

2.4.2 Intégrité des données agrégées [6, 27]

La section précédente a considéré le problème de l'agrégation des données chiffrées. Cette section traite du problème, au moins aussi difficile, de l'integrité des données agrégés. Les données sont agrégées lors de leur acheminement vers le puits. Mais comment le puits peut-il avoir la certitude que l'agrégat qu'il a reçu a été généré à partir de données légitimes (c'est à dire émis par des capteurs autorisés) ? Ce problème est difficile car le puits, bien évidemment, ne reçoit pas les valeurs individuelles et que, comme nous l'avons démontré [27], les étiquettes d'authentification (les tags MAC) ne sont agrégables. Perrig et al. ont récemment proposé une solution utilisant des arbres de Merkle. Cependant cette solution nécessite au moins 3 échanges de messages et est donc coûteuse en terme de bande passante et énergie. De plus, elle nécessite que le réseau soit stable entre les différents échanges de messages, ce qui est une hypothèse peu réaliste pour les réseaux de capteurs.

L'objectif principal de nos travaux de recherche était ici de proposer une solution qui : (1) permette au puits de vérifier l'intégrité de l'agrégat en *un seul échange de message*, (2) autorise le chiffrement des données et (3) soit efficace en terme de bande passante et de consommation d'énergie. Nous avons alors développé un nouveau protocole d'agrégation, *ABBA*, qui non seulement satisfait les conditions précédentes, mais permet d'étendre l'agrégation des données à d'autres opérations que l'addition.

Notre solution utilise un approche de type "bins and balls" : l'espace possible des valeurs mesurées est divisé en n cases. Chaque case correspond à un interval de valeurs. Si, par exemple, les capteurs mesurent des températures qui peuvent varier entre $[-100, +100]$, et que $n = 10$, alors la première case correspond aux valeurs comprises dans l'interval $[-100, -80]$, la deuxième $[-81, -60]$, etc. Lorsqu'un capteur effectue une mesure t, au lieu d'envoyer cette valeur au puits, il incrémente la case correspondante à la valeur t. Par exemple, si $t = 25$, il incrémente la case correspondant à l'interval $[20, 40]$. Chaque case est ensuite chiffrée en utilisant la fonction homomorphique décrite précédemment, et l'ensemble est envoyé à son père. Le père ajoute les messages qu'il reçoit de ses fils, case par case, incrémente la case correspondant à sa mesure, chiffre le tout et envoie le résultat à son père. Lorsque le puits reçoit l'agrégat, il peut déchiffrer chaque case et ainsi retrouver la valeur de chacune d'entre elles. Le puits reçoit alors une distribution des valeurs mesurées par les capteurs.

Ce nouveau protocole permet d'assurer la confidentialité et permet au puits d'obtenir suffisamment d'information pour calculer un grand nombre d'agrégats (moyenne, median, min/max,...). *Mais comment fournit-elle l'intégrité des données ?*

Note que lorsque le puits récupère l'ensemble des cases, il peut s'assurer que la somme de leur valeur est égale à n. Si cette valeur est différente de n, alors l'agrégat a été modifié : un attaquant a ajouté ou soustrait des valeurs à certaines cases ! Un attaquant intelligent pourrait soustraire p à un ensemble de cases et ajouter p à d'autres cases pour modifier la distribution de l'agrégat. Dans ce cas, la somme de la valeur des cases serait égale à n. Cependant, nous avons montré par nos travaux, que si l'attaquant est trop agressif, c'est à dire qu'il utilise un grand p, la probabilité qu'une des cases ait une valeur inférieure à zéro (valeur incohérente) est importante. L'attaquant qui ne connait pas les valeurs des cases (car elles sont chiffrées) travaille à "l'aveugle". Le puit, lui, pourra alors détecter la modification. La sécurité fournit par notre solution est probabiliste : un attaquant aura une probabilité non-négligeable de réussir son attaque s'il ne pas trop agressif. Cependant l'impact de son attaque sera négligeable dans ce cas. Si l'attaquant veut modifier la distribution de façon significative, par exemple en déplacant la moyenne de quelques unités, la probabilité que son attaque aboutisse devient alors très faible.

Les résultats de ce travail sont décrits dans un article qui a été publié à WiOpt 2008 [6]. Cet article détaille notre protocole et analyse sa sécurité de façon analytique. Il présente également quelques résultats de simulation.

2.5 Virus et Vers Informatiques pour les réseaux de capteurs [35]

Les réseaux de capteurs se développant et faisant partie des infrastructures critiques, il est tout à fait naturel de penser à la menace que posent les virus et vers sur ces nouveaux réseaux.

Dans l'Internet, un attaquant peut compromettre des machines en exploitant des vulnérabilités, résultant souvent d'un dépassement d'un buffer en mémoire. Un capteur étant un micro-ordinateur, possédant un CPU, de la mémoire, des Entrées/Sorties, à priori, tout peut nous faire penser que ces types d'attaque y sont transposables. Cependant, les capteurs ont plusieurs caractéristiques qui rendent leur compromission à distance par un virus très délicat :

- Les mémoires "programmes" et "données" sont souvent physiquement séparées (mémoire FLASH pour le programme, et mémoire SRAM pour les données et la pile) et utilisent des espaces d'adressage différents. Il est alors souvent impossible d'éxécuter du code qui serait inséré dans la pile, comme c'est souvent le cas dans les attaques qui exploitent un dépassement de buffer pour écraser la pile (Stack-based buffer overflow).
- Le code application est souvent protégé en écriture. Un attaquant ne peut pas modifier les programmes présents en mémoire.
- La taille des paquets que peut recevoir un capteur est très limitée (typiquement 28 octets), ce qui rend l'injection de code "utile" difficile.

Les techniques qu'utilisent les vers pour compromettre une machine sur Internet, ne peuvent donc pas être utilisées directement sur les capteurs. Nous avons cependant montré, en concevant un des premiers virus/vers pour capteurs de type Micaz/TinyOS, que la conception de virus, bien que difficile, n'est pas impossible.

Nous avons utilisé, pour arriver à notre objectif, deux propriétés que possèdent souvent le réseaux de capteurs :

- un réseau de capteurs est très souvent homogène, c'est à dire composé de dispositifs similaires, configurés avec les mêmes composants. La configuration mémoire de tous les capteurs est donc souvent identique. En compromettant un noeud, un attaquant peut facilement identifier le code présent en mémoire de l'ensemble des noeuds du réseau.
- Chaque capteur doit souvent être reconfigurable à distance après déploiement, au cas ou un bug doit être corrigé ou un autre programme doit être chargé en mémoire. Cette reconfiguration est souvent réalisée par un logiciel (par exemple Deluge sous TinyOS), préalablement installé sur le capteur, qui copie le nouveau programme de la mémoire RAM externe vers la mémoire exécutable.

Le virus que nous avons conçu opére comme suit :

– une vulnerabilité dans le programme est exploitée en envoyant un paquet, formaté de façon adéquate, qui écrit, par un dépassement de buffer, dans la pile. Ce dépassement de buffer est utilisé pour exécuter une série de groupes d'instructions qui vont copier un octet du paquet vers une zone mémoire inutilisée. Plus spécifiquement, le premier groupe d'instructions configure les registres (en utilisant les données qui sont dans la pile et qui ont éte écrasés par le paquet lors du dépassement de la pile), qui vont permettre au deuxième groupe d'instructions de copier l'octet qui se trouve dans le paquet vers la position en mémoire qui aura été choisie. Le paquet doit être convenablement formatté afin de contenir les adresses des intructions à exécuter et les valeurs des registres à configurer.
– Le paquet précédent permet de copier un octet envoyé sur la mémoire donnée du capteur. En envoyant plusieurs paquets de ce type, nous pouvons créer une fausse pile en mémoire. Cette fausse pile sera écrite dans une zone mémoire qui n'est pas utilisée par le proprramme car elle est localisée au delà des zones *data* et *bss* et en dessous de la valeur maximum de la pile. De plus cette zone mémoire n'est pas effacée lors d'un reboot. Cette fausse pile contient des données qui permettent de configurer les registres utilisés par les instructions appelées lors de l'étape suivante, ainsi que le code malveillant que l'on veut insérer dans le capteur.
– Une fois la fausse pile insérée en memoire, il suffit alors d'envoyer un dernier paquet qui, en exploitant la même vulnérabilité, va : (1) exécuter un groupe d'instructions qui redirige le

pointeur de pile (stack pointeur) vers la fausse pile, (2) lancer un groupe d'instructions qui configure les registres pour le dernier groupe, qui (3) copie le code malicieux en mémoire exécutable.
- Un dernier paquet peut alors lancer l'exécution du code malicieux.

Il faut noter qu'après chaque étape, le capteur est rebooté en retournant à l'adresse exécutable 0. Le code malicieux peut lui-même lancer la même attaque sur les voisins du capteur compromis et transformer le virus en vers. Les détails de ce travail sont décrits dans un article qui est en cours de soumission [35].

2.6 Conclusions

La sécurité des réseaux de capteurs a été un de nos plus importants domaines de recherche ces dernieres années. Nos contributions sont à la fois théoriques, avec le développement de nouveaux algorithmes de chiffrement et de génération de nombres aléatoires, et pratiques, avec la conception de nouveaux protocoles d'échange de clés et de virus/vers.

Toutes nos résultats sont implantés sous TinyOS et intégrés dans une librairie qui est développée dans la cadre du projet Européen UbiSec&Sens. Cette librairie a pour objectif d'aider les développeurs à sécuriser leurs applications.

Aujourd'hui les réseaux de capteurs sont essentiellement déployés dans des laboratoires à des fins de recherche. Des déploiements commerciaux devraient cependant voir le jour prochainement. Ce domaine de recherche devrait alors se développer considérablement, d'autant plus qu'il reste encore beaucoup de problèmes scientifiques à résoudre. Par exemple, dans beaucoup d'applications, il est nécessaire que le puits connaisse la localisation géographique des noeuds du réseau. Les protocoles de localisation qui existent ne garantissent pas une localisation correcte en présence d'un attaquant actif. La sécurisation des protocoles de localisation est un sujet de recherche très difficile qui mérite réflexion et sur lequel nous planifions de travailler. D'autres sujets importants sont la détection d'intrusion, la sécurisation des systèmes d'exploitation, la sécurisation des protocoles de mise à jour de code, etc.

Chapitre 3

Sécurité des systèmes RFID

3.1 Introduction

3.1.1 Les technologies RFID

L'identification par radio fréquence, connue sous le nom de RFID (Radio Frequency Identification) est devenue aujourd'hui une technologie incontournable. Cette technologie, qui permet d'identifier et parfois authentifier un objet à distance, est devenue omniprésente. Elle est utilisée pour la traçablité dans les chaînes logistiques (en remplacement des codes barres), dans les passeports électroniques, les systèmes de paiement à carte sans fil, les badges d'accès, les clés des voitures. Elle est également utilisée, dans certains cas, pour identifier des personnes (RFID implantable) ou des animaux.

Un système RFID est constitué de 3 éléments : une étiquette, un lecteur et une base de donnée (voir figure 3.1).

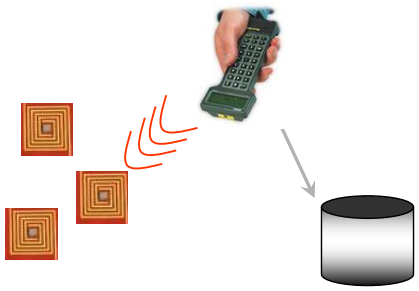


Fig. 3.1 – Système RFID : étiquettes, lecteur et base de données

L'étiquette est un petit circuit électronique, équipé d'une antenne, qui stocke en mémoire un ensemble de données (typiquement un ou plusieurs identifiants). Elle répond à une requête émise par un lecteur en diffusant son identifiant. Le lecteur transmet, en temps réels ou en différé selon l'application, cet identifiant à la base de donnée pour traitement. La base de donnée peut alors identifier l'objet ou la personne associée à l'étiquette.

Il existe une large variété d'étiquettes possédant des caractéristiques très différentes. Certaines sont très bon marché, minuscules et ne sont dotées que d'une petite mémoire, d'autres sont plus onéreuses et possèdent des capacités de calcul importantes. Certaines étiquettes, comme celles utilisées dans les passeports électroniques, peuvent même effectuer des opérations complexes et

coûteuses, tel que des opérations de cryptographie à clé publique. Les étiquettes sont généralement classées en deux catégories : les étiquettes *actives* et les étiquettes *passives*. Les étiquettes actives possèdent une source d'énergie interne (généralement une pile). Ces étiquettes sont onéreuses et volumineuses. Elles sont généralement utilisées pour les applications qui nécessitent des capacités de calcul ou de transmission importantes.

Les étiquettes passives ne possèdent aucune source d'énergie et sont alimentées par le champ électromagnétique émis par le lecteur. L'antenne de l'étiquette capte certaines fréquences qui lui fournissent suffisamment d'énergie pour lui permettre d'émettre à son tour son identifiant. Ils ont une portée de communication plus faible que les étiquettes actives : quelques centimètres pour les étiquettes fonctionnant en basse (125kHz) ou haute fréquence (13,56 MHz), et quelques mètres en ultra haute fréquence (900 MHz).

Les étiquettes passives les plus utilisées aujourd'hui sont les étiquettes EPC (Code Produit Electronique). Elles sont très peu onéreuses (quelques centimes) et ne sont dotées que d'une mémoire accessible en lecture, qui contient un identifiant unique de 128 bits. Elles peuvent dans certains cas effectuer des opérations reposant sur de la logique cablée. Les étiquettes EPC sont essentiellement utilisées dans les chaînes logistiques, et particulier dans la grande distribution.

3.1.2 Sécurité des systèmes RFID

De façon générale, les applications RFID peuvent être classées en deux grandes catégories : les applications dont l'objectif est d'améliorer la sécurité (badges d'accès de tout type, cartes de paiement) et les applications dont l'objectif principale est d'identifier/tracer des objets ou des personnes (logistique, suivi/surveillance d'animaux,...). Ces deux types d'applications ont des contraintes de sécurité très différentes. Dans le premier cas, l'étiquette RFID est utilisée comme un moyen *d'authentification* : l'étiquette doit non seulement s'identifier, c'est à dire réveler son identité, mais aussi la prouver au lecteur. L'identification est, dans ce cas, insuffisante, car un attaquant pourrait simplement écouter l'identifiant, puis le rejouer plus tard pour usurper l'étiquette.

Il existe aujourd'hui deux types d'authentication : l' *authentification faible* et l'*authentification forte*. Les mécanismes d'authentification faible sont relativement peu sûres : une étiquette répond à une requête d'un lecteur en envoyant son identifiant public accompagné d'un secret, uniquement connu de l'étiquette et de sa base de donnée. Il suffit alors à un attaquant d'écouter un échange pour apprendre le secret et le rejouer par la suite. Les systèmes RFID ne peuvent apporter un niveau de sécurité acceptable uniquement si un mécanisme d'authentification forte est utilisé. Dans les systèmes à authentification forte, chaque noeud est configuré avec une clé secrète, qu'il utilise pour répondre à un défi (challenge). Cette réponse est soit une signature du défi, son chiffrement avec la clé secrète ou sa valeur hachée (en utilisant une fonction de hachage concaténé avec la clé par exemple). Dans ce cas, étant donné que la réponse varie en fonction du défi, il devient difficile pour un attaquant de générer la réponse correcte sans connaître le secret. Le mécanisme d'authentification nécessite donc des fonctions cryptographiques (chiffrement, signature, hachage). Par conséquent, les étiquettes qui fournissent cette fonctionalité sont généralement plus onéreuses que les étiquettes de type EPC, qui ont pour unique objectif d'identifier des objets.

Pour les étiquettes à très bas coût, de type EPC, le problème de sécurité le plus délicat est le problème de la *traçabilité malveillante*. En effet, étant donné qu'une étiquette répond toujours avec son identifiant, et que cet identifiant ne change pas, elle peut être utilisée pour tracer un objet ou une personne. Si cette étiquette est attachée à une chemise, elle peut fournir beaucoup d'information sur les activités du porteur de la chemise (voir figure 3.2). Ce problème est moins important avec les étiquettes haut de gamme, qui peuvent, par exemple, chiffrer leur identifiant avant de l'envoyer. Il est très délicat avec les étiquettes EPC qui possède des capacités de calcul très faibles.

Le problème de la traçabilité malveillante est crucial. Les systèmes RFID pourront difficilement être déployés si ce problème n'est pas résolu. Benetton et Wallmart en ont fait les frais lorsque des défenseurs des libertés individuelles, ont appelé à boycotter ces deux marques qui voulaient équiper leurs produits d'étiquettes RFID EPC.

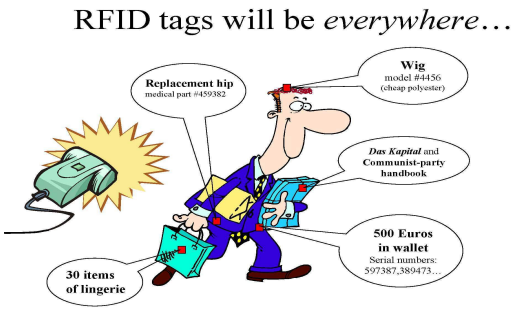


FIG. 3.2 – Traçabilité Malveillante : Illustration

3.2 Identification Secrète des Etiquettes

Nos travaux se sont essentiellement portés sur le problème d'identification secrète des étiquettes. Notre objectif était de concevoir des mécanismes et protocoles qui permettent de rendre la traçabilité de ces étiquettes plus difficile, tout en utilisant des étiquettes EPC standards.

Nous avons proposé deux solutions : les *étiquettes brouilleuses* (noisy tags) et une solution, appelée *ProbId*. La première solution, très efficace en terme de calcul et mémoire, permet une identification secrète des étiquettes dans environnements surveillés (banques, entreprises, aéroports,...) et protègent contre des attaquants passifs. La deuxième solution est plus générique et protège les étiquettes contre les attaques passives et actives. Elle est cependant un peu plus onéreuse car, bien que ne nécessitant que très peu de calcul, elle utilise une zone mémoire ROM de quelques centaines de bits.

Ces deux solutions sont résumées dans le reste de ce chapitre.

3.2.1 Les Etiquettes Brouilleuses (Noisy Tags) [13]

La solution basée sur les étiquettes brouilleuses permet à un lecteur et à une étiquette d'échanger un secret sans utiliser de fonctions cryptographiques. Ce secret peut être ensuite utilisé pour chiffrer l'identifiant (en faisant un *ou* exclusif -xor- du secret et de l'identifiant).

L'idée de base de ce protocole est la suivante : une entité A peut envoyer un secret à une autre entité B, si B brouille le canal de transmission pendant la transmission. B pourra alors supprimer le bruit, qu'il a généré, du signal de réception et retrouver le secret. Tout autre noeud qui écoute la transmission n'entendra que du bruit car il ne pourra pas différencier le bruit généré par B du secret transmit par A.

Notre proposition utilise cette idée et l'applique aux étiquettes RFID. Elle utilise des étiquettes brouilleuses qui sont déployées dans l'environnement du lecteur (au guichet d'un banque, dans un aéroport,...) et qui génèrent le bruit sur le canal de transmission. Ces étiquettes sont des étiquettes EPC qui implémentent un fonction de hachage (MD5, par exemple). Chacune de ces étiquettes, EB_i, est configurée avec une clé secrète, k_i, partagée avec le lecteur.

Description du Protocole NTP **(Noisy Tag Protocol)** : Le protocole de base, lorsqu'une étiquette $\mathcal{T}$ passe devant le lecteur, fonctionne comme suit (voir figure 3.3) :

- Le lecteur envoie un requête qui contient une valeur aléatoire, $Nonce$.

- L'étiquette $\mathcal{T}$ répond avec un bit secret (ce bit peut être aléatoire ou peut appartenir à un clé stockée préalablement en mémoire).
- Chaque étiquette brouilleuse, EB_i, répond avec un bit, b_i, qui est calculé à partir d'un fonction de hachage $h(.)$, la valeur $Nonce$ et le secret k_i. Par exemple, b_i peut correspondre au bit de poids faible du résultat de $h(k_i|Nonce)$.
- Le lecteur reçoit donc 1 bit de chacune des étiquettes brouilleuses et le bit secret émis par le étiquette $\mathcal{T}$. Comme il connaît, les clés k_j et la valeur Nonce, il peut calculer les bits envoyés par chaque étiquette brouilleuse, et identifier le bit envoyé par le tag $\mathcal{T}$.

En répétant ce protocole n fois, l'étiquette $\mathcal{T}$ peut transmettre un secret de n bits au lecteur. Un espion, qui écoute sur le canal radio, entend un ensemble de bits. Mais ne connaissant pas les clés k, il ne peut pas identifier les bits envoyés par les étiquettes brouilleuses et retrouver le bit secret.

L'hypothèse sous-jacente à ce protocole est qu'un espion ne peut pas identifier la source de bits en utilisant, par exemple, la puissance de transmission. Cette hypothèse est validée en utilisant un nombre important d'étiquettes brouilleuses.

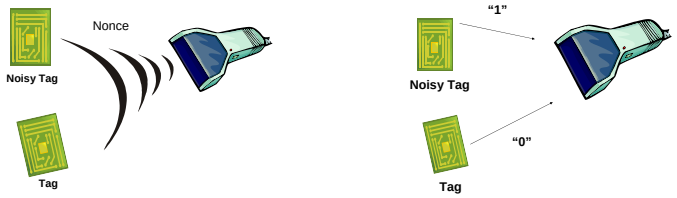


FIG. 3.3 – Noisy Tag Protocol : Le lecteur diffuse un Nonce. Le tag $\mathcal{T}$ répond avec le bit secret. Le noisy tag répond avec un bit qui est le bit de point faible de $md5(nonce|ki)$.

Discussions L'article scientifique [cast06a] présente trois variantes de ce protocole de façon plus détaillée. La solution que nous proposons permet à une étiquette d'envoyer un secret à un lecteur. Elle est très bien adaptée aux étiquettes bon marché, tel les EPCs, car ne nécessite aucune opération sur les étiquettes. Seules les étiquettes brouilleuses doivent implanter une fonction de hachage.

Une des limitations de cette solution est qu'un attaquant peut disposer ces propres étiquettes brouilleuses et interroger l'étiquette $\mathcal{T}$ avec un lecteur. L'étiquette $\mathcal{T}$, n'authentifiant pas le lecteur, repondra alors à cette requête en envoyant son secret et son identifiant. L'attaquant pourra alors identifier l'étiquette. Notre solution est donc plutôt adaptée aux environnements surveillés (magasins, banques, aéroports, bus, ...) dans lesquels la présence d'un attaquant actif est détectable.

En résumé, notre protocole est sûre contre un attaquant passif (espion) et contre un attaquant actif dans un environnement surveillé. Dans les environnements ouverts, elle peut être associée à la solution "tag blocker" de RSA qui brouille les identifiants émis par les étiquettes. Avec cette solution, aucun lecteur ne peut identifier les étiquettes qui se trouvent sous la protection du "tag blocker". L'utilisateur pourra alors désactiver le "tag blocker" dans les environnements surveillés s'il le juge nécessaire.

3.2.2 Identification Probabiliste [23]

Le protocole $ProbIP$ (Probabilistic Identification protocol) propose une nouvelle approche au problème de l'identification secrète des étiquettes RFID. Comme la solution précédente, $ProbIP$ ne nécessite aucun calcul de la part des étiquettes et est, par conséquent, bien adaptée aux étiquettes de type EPC.

La différence avec la solution précédente est qu'elle ne fait aucune hypothèse sur les propriétés physiques des étiquettes et est sûre aussi bien contre les attaques passives que les attaques actives. Sa sécurité repose sur un problème NP-complet.

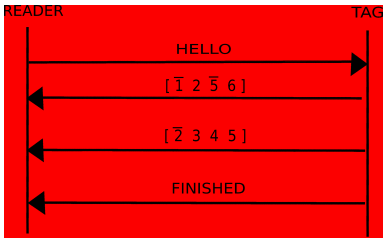
Avec $ProbIP$, chaque étiquette $\mathcal{T}_i$ est configurée avec une clé secrète de longueur K, k_i. Cette clé est utilisée comme un vecteur de bits, où $k_i[1]$ est le premier bit, $k_i[1]$ le deuxième bit, etc... Le lecteur (ou le serveur de base de données), $\mathcal{R}$, est, lui, configuré avec la clé des n étiquettes du systèmes.

Description du Protocole : Le protocole, entre l'étiquette $\mathcal{T}_j$, et le lecteur $\mathcal{R}$, opère comme suit :

1. $\mathcal{R}$ initialises une identification en diffisant un message `HELLO`.
2. A la réception de ce message `HELLO`, $\mathcal{T}_j$ répond avec P *paquets* et un message `FINISHED`, où P is un paramètre du système. Un paquet est une liste de $2L$ valeurs, $a_1, b_1, a_2, b_2 \ldots, a_L, b_L$, où a_i est un index aléatoire, choisi entre 1 et K, et b_i sont des bits aléatoires qui satisfont l'équation :
$$\sum_{i=1}^{L} k_j[a_i] \oplus b_i = L/2 \tag{3.1}$$
3. A la réception de chaque paquet, $\mathcal{R}$ calcule le résultat de l'équation 3.1 avec les clés de tous les étiquettes qu'il a en mémoire. Les étiquettes dont la clé qui satisfont l'équation, font alors partis des candidats potentiels. Chaque paquet réduit la liste des candidats potentiels. Au bout de P paquets, un seul candidat reste : il s'agit de l'étiquette à identifier.

Illustration : Prenons un exemple pour illustrer notre protocole. Considérons un système qui utilise les paramètres suivants : $L = 4$, $K = 6$ et $n = 4$. $\mathcal{T}_1$ est configuré avec la clé k_1 =`011001`, $\mathcal{T}_2$ avec la clé k_2 =`100101`, $\mathcal{T}_3$ avec la clé k_3 =`011110` et finalement $\mathcal{T}_4$ avec k_4 =`001110`.

Faisons l'hypothèse que l'étiquette à identifier est $\mathcal{T}_2$. Le protocole opère comme suit (voir figure 3.2.2).



1. $\mathcal{R}$ diffuse un message `HELLO`.
2. $\mathcal{T}_2$ envoie deux paquets et le message `FINISHED`. Le premier paquet est défini par `[`$\bar{1}$ `2` $\bar{5}$ `6]`, pour lequel l'équation 3.1 avec k_2 est $(1 \oplus 1) + (0 \oplus 0) + (0 \oplus 1) + (1 \oplus 0) = 2 = L/2$. Le deuxième paquet est défini par `[`$\bar{2}$ `3 4 5]` pour lequel l'équation 3.1 avec la clé k_2 est $(0 \oplus 1) + (0 \oplus 0) + (1 \oplus 0) + (0 \oplus 0) = 2 = L/2$. Nous utilisons la notation suivante : $\bar{a}_i$ si $b_i = 1$ et a_i si $b_i = 0$
3. A la réception du premier paquet, le lecteur calcule pour chaque des 4 étiquettes l'équation 3.1. Le résultat de cette équation est 4 pour $\mathcal{T}_1$, 2 pour $\mathcal{T}_2$, 2 pour $\mathcal{T}_3$ et 1 pour $\mathcal{T}_4$. Les deux seuls candidats sont donc $\mathcal{T}_2$ et $\mathcal{T}_3$.
4. A la réception du second paquet, le lecteur calcule pour $\mathcal{T}_2$ et $\mathcal{T}_3$ l'équation 3.1. Le résultat de cette équation est 2 pour $\mathcal{T}_2$ et 3 pour $\mathcal{T}_3$. Le seul candidat restant est alors $\mathcal{T}_2$, l'étiquette à identifier.

Discussions : Ce protocole permet à un lecteur autorisé d'identifier une étiquette de façon secrète. En d'autres termes, un espion qui écoute les échanges entre une étiquette et un lecteur ne peut pas identifier, ou mieux encore lier deux itérations du protocole entre eux, s'il ne possède pas la clé de l'étiquette.

Plus spécifiquement, nous avons montré que la sécurité de ce protocole peut-être ramener à résoudre un problème NP-complet (L/2-in-L LSAT). La seule solution pour résoudre ce problème est d'utiliser un résolveur LPBC (Linear Pseudo-Boolean Constraint). Nous avons utilisé un tel résolveur et avons évaluer le nombre d'échanges étiquette-lecteur qu'un espion doit écouter pour pouvoir retrouver la clé secrète de l'étiquette en un temps raisonnable. Les résulats complets sont présentés dans notre article scientifique [23].

La seule solution prouvée sûre qui existe aujourd'hui repose sur l'utilisation de pseudonymes : chaque étiquette est équipée de m identifiants pseudonymes et répond à chaque requête en utilisant un identifiant différent. En écoutant, m itérations de ce protocole, un espion peut collecter la liste des pseudonymes et écouter les futures identifications de cette étiquettes. Si un système RFID supporte $n = 10^7$ étiquettes, et que chaque étiquette est configurée avec 15 identifiants, une mémoire de taille $15 * \lceil \log_2(10^7 * 15) \rceil = 420$ bits est nécessaire. Dans ce cas, la sécurité d'une l'étiquette est complétement inexistante au bout de 16 itérations du protocole entre l'étiquette et le lecteur. Avec les mêmes paramètres, $ProbIP$ fournit une sécurité plus forte : un espion qui a écouté 16 itérations du protocole doit, en plus, effectuer des calculs très complexes (dont la temps d'éxécution a été évalué à 1.46e28 seconds sur un machine avec un processeur Pentium-D de 3 GHz) pour retrouver la clé de l'étiquette. Comme montré dans notre article, il existe un compromis entre le nombre de calculs et le nombre d'itérations à écouter : plus l'espion posséde d'itérations du protocole, plus facile il lui sera de calculer la clé secrète.

3.3 Conclusions

Par nos travaux, nous avons montré qu'il était possible d'améliorer la sécurité des étiquettes EPC sans les modifier. Cependant nous devons constaté que nos solutions ne sont pas infaillibles. Un attaquant obstiné et possédant un certain budget pourra certainement arriver à ces fins. Avec le protocole NTP (Noisy Tag), un attaquant équipé d'un système d'écoute élaboré pourra peut-être identifier les sources des messages et identifier ceux émis par les étiquettes brouilleuses. Il pourra alors retrouver le secret échangé. Avec le protocole $ProbIP$, un attaquant obstiné pourra, en écoutant suffisamment d'itérations, arriver à ces fins.

Ces travaux ont permi d'explorer les limites des solutions qui n'utilisent aucune fonction cryptographique. Il semble qu'il est difficile (peut-être impossible) d'arriver à une solution aussi sûre que les solutions cryptographiques. Cependant des solutions à bas-coût, telle ProbIP, sont appropriées pour des étiquettes EPC qui sont attachées à des objets dont la durée de vie est limitée (bouteilles de lait, savons,...) ou dont le nombre d'interactions avec un lecteur peut être limité et controllé. Par ailleurs, la sécurité de la solution $ProbIP$ est tout à faite satisfaisante, si les clés des étiquettes peuvent changées périodiquement (par exemple tous les 1000 identifications).

Nos travaux de recherche continuent et nous ne désespéront pas de trouver des solutions qui fournissent une sécurité équivalente aux solutions qui utilisent des fonctions cryptographiques. L'avenir nous le dira...

Chapitre 4

Conclusions et Perspectives

Pour conclure ce mémoire je présente quelques axes de recherche que je souhaite développer dans les quatre années à venir.

La sécurité de l'Internet des choses

Mes travaux de recherche dans le domaine des systèmes sans-fil ont, jusqu'à présent, principalement porté sur la sécurité des systèmes sans fil autonomes, c'est à dire non connectés à l'Internet. J'ai étudié la sécurité des réseaux ad-hoc mobiles, des réseaux de capteurs et des systèmes RFID en faisant l'hypothèse qu'ils n'étaient pas forcément connectés à l'Internet. Cependant, il est fort à penser que tous ces appareils seront un jour ou l'autre connecté à l'Internet et seront accessibles à distance. Cette "Internet des choses" permettra de développer de nouvelles applications qui seront à la base de nouveaux problèmes de sécurité.

Un utilisateur de l'internet pourra accéder à distance à des capteurs ou lire des étiquettes RFID. Il pourra alors régler à distance la température de sa maison, ou vérifier, en temps réel, l'état d'enneigement avant une randonnée. Les personnes agées ou malades pourront être équipées de capteurs médicaux et être surveillées à distance par des médecins, infirmiers ou des proches.

Cet Internet des choses qui constitue incontestable un progrès doit toutefois être développé et mis en place avec précaution. En effet, toutes ces nouvelles capacités de surveillance peuvent être abusées par un utilisateur malveillant pour suivre les activités et mouvements des porteurs de ces capteurs à leur insu. Tout capteur, même anodin, peut fournir beaucoup d'information à un adversaire. Il a été montré que des capteurs de température installés dans une pièce peuvent être utilisés pour déterminer à distance si la pièce est occupée ou pas. De même, de plus en plus d'appareils médicaux, tels que défribillateurs ou pace-makers, sont équipés d'interface sans fil avant de pouvoir être configuré par un médecin sans intervention chirurgicale. Ces nouveaux appareils peuvent être abusés par un utilisateur mal-intensionné afin de lire des informations confidentielles ou pire encore dérégler l'appareil médical [12] !

Il convient donc d'étudier avec précaution les problèmes de sécurité et de protection de la vie privée généré par cet Internet de capteurs. Noter que sécuriser les liens de suffit pas. Dans certain cas, le fait de détecter une communication sans fil peut révéler beaucoup d'information. Il peut, par exemple, réveler qu'un utilisateur est porteur d'un appareil médical ce qui peut être une information utile pour un employeur ou un assureur. Il convient donc de concevoir des communications invisibles qui ne soient détectables uniquement par des récepteurs autorisés. Tout un programme...

La sécurité de l'Internet de demain

L'Internet est victime de son succès. Bien qu'il soit encore très opérationnel, il est de plus en plus admis dans la communauté scientifique que l'Internet d'aujourd'hui a atteint ses limites et qu'il faudra bientôt le remplacer ou en tout cas le faire évoluer très fortement. L'idée de l'Internet

du future est donc lancée et génère déjà beaucoup de travaux, discussions, débats et polémiques. Certains chercheurs prônent une refonte complète des mécanismes et architecture de l'Internet. Ce sont les partisans de l'approche "Clean Slate" (ardoise propre). D'autres chercheurs, plus raisonnables à mon avis, prône un évolution de l'Internet et l'intégration de nouveaux mécanismes qui permettent de résoudre les limitations actuelles, sans remettre en cause les protocoles de base.

D'ailleurs quelles sont les limitations de l'Internet aujourd'hui ? Différentes limitations existent mais la plus importante, aujourd'hui, semble être la sécurité, ou plus particulièrement le manque de sécurité de l'Internet. L'Internet a été conçu, il y a une trentaine d'année par l'armée américaine, comme un réseau fermé. L'objectif principal était de concevoir un réseau qui reste opérationnel même si plusieurs routeurs s'arrêtaient de fonctionner, par exemple à la suite d'un bombardement. Les attaques internes n'étaient pas vraiment prises en compte car l'accès au réseau était alors très controllé et uniquement quelques priviligiés pouvaient l'utiliser.

La situation est différente aujourd'hui. Presque tout le monde est connecté à l'Internet et les ennemis sont parmi nous. En d'autres termes, les attaquants sont maintenant internes. L'Internet n'est pas adapté à ces nouvelles attaques et nécessite d'être adapté/amélioré. Je souhaiterai contribuer à cette réflexion et contribuer à cette évolution de l'Internet.

Un des problèmes qui existent pour atteindre cet objectif est de comprendre comment l'Internet est exploité par les cyber-déliquants et comment prévenir ces attaques. Les cyber-délinquants sont très créatifs, très dynamiques et de plus en plus compétents. Alors qu'il y a seulement quelques années, les attaques étaient essentiellement menées par des adolescents en manque de reconnaissance qui exécutaient des scripts récupérés sur l'Internet, elles sont aujourd'hui lancées par des professionnelles dont l'objectif est pécunier. Les attaques deviennent de plus en plus complexes, structurées, variées et ciblées. Les cyber-délinquants utilisent des techniques très élaborés et à la pointe de la technologie (voire de la recherche), tel que les systèmes Pair-à-Pair, les rootkits ou le DNS fast-fluxing, pour améliorer la fiabilité de leurs attaques et dissimuler leurs actions.

Nos travaux sur les robots de machines compromises (les botnets) ont montré l'ampleur du phénomème. Un botnet est un réseau de machines qui ont été compromises et qui sont sous le contrôle d'un cyber-délinquant. Ces machines ont souvent un comportement normal car tout est fait pour qu'une machine compromise reste opérationnelle le plus longtemps. Nous avons réussi à infiltrer le botnet Storm et montré qu'il était constitué de 6000 à 7000 machines actives, prêtes à lancer à tout moment des attaques (envoi de pourriels, attaque de déni de service) sur l'ordre de leur maître.

Une vrai économie souterraine, très bien structurée, s'est crée. Certains personnes trouvent des failles dans des logicielles, qu'ils revendent à d'autres personnes qui les utilisent pour créer des exploits. Ces exploits sont ensuite vendus à des cyber-déliquants qui les utilisent pour compromettre des machines afin créer des botnets. Ces botnets sont ensuite "loués" pour envoyer des pourriels ou récupérer des informations secrètes (tels que des mots de passe, numéro de compte bancaires,...) qui sont elles-même revendues.

Mon objectif est, dans un premier temps, de comprendre les mécanismes et rouages utilisés par les cyber-délinquants afin de contribuer aux travaux sur l'architecture de l'Internet du future. Il est probable que les problèmes de cyber-sécurité ne pourront être résolus par des solutions techniques. Il faudra certainement adopter une approche pluridisciplinaire qui combine des solutions techniques, légales et économiques.

Bibliographie

[1] I. Aad and C. Castelluccia. Introducing service differentiation into IEEE. In *Proceedings of the Fifth IEEE Symposium on Computers and Communications*, 2000.

[2] I. Aad and C. Castelluccia. Enhancing IEEE 802.11 performance in congested environments. *Annales des télécommunications*, 2003.

[3] I. Aad, C. Castelluccia, and J.P. Hubaux. Packet coding for strong anonymity in ad hoc networks. In *IEEE SECURECOMM 2006.*, 2006.

[4] Imad Aad and Claude Castelluccia. Differentiation mechanisms for IEEE 802.11. In *INFOCOM*, pages 209–218, 2001.

[5] Imad Aad and Claude Castelluccia. Remarks on per-flow differentiation in IEEE 802.11. In *European Wireless 2002*, 2002.

[6] Castelluccia and C. Soriente. ABBA : A balls and bins approach to secure aggregation in WSNs. In *Sixth International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'08)*, Berlin, Germany, 2008.

[7] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from CA-oblivious encryption. In *In Advances in Cryptology - ASIACRYPT 2004*, 2004.

[8] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from CA-oblivious encryption (short paper). In *ACM Symposium on Principles of Distributed Computing (PODC)*, 2004.

[9] C. Castelluccia and A. Spognardi. ROK : A robust key pre-distribution protocol for multi-stage wireless sensor networks. In *IEEE SECURECOMM*, September 2007.

[10] Claude Castelluccia. Cryptographically generated addresses for constrained devices. *Kluwer Wireless Personal Communications special issue on Wireless Security for Next Generation Communications*, 29(3-4), 2004.

[11] Claude Castelluccia. Securing very dynamic groups and data aggregation in wireless sensor networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Networks (MASS 2007)*, Pisa, Italy, October 2007.

[12] Claude Castelluccia. Des cardiaques victimes d'attaques informatiques. *La Recherche*, May 2008.

[13] Claude Castelluccia and Gildas Avoine. Noisy tags : A pretty good key exchange protocol for RFID tags. In Josep Domingo-Ferrer, Joachim Posegga, and Daniel Schreckling, editors, *CARDIS*, volume 3928 of *Lecture Notes in Computer Science*, pages 289–299, Tarragona, Spain, April 2006. IFIP, Springer-Verlag.

[14] Claude Castelluccia, Francis Dupont, and Gabriel Montenegro. A simple privacy extension for mobile ipv6. In *MWCN*, pages 239–249, 2004.

[15] Claude Castelluccia, Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Secure acknowledgment aggregation and multisignatures with limited robustness. *Computer Networks*, 50(10) :1639–1652, 2006.

[16] Claude Castelluccia and Gabriel Montenegro. Protecting AODV against impersonation attacks. *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, 2002.

[17] Claude Castelluccia and Gabriel Montenegro. Securing group management in IPv6 with cryptographically generated addresses. In *ISCC '03 : Proceedings of the Eighth IEEE International Symposium on Computers and Communications*, Washington, DC, USA, 2003. IEEE Computer Society.

[18] Claude Castelluccia, Gabriel Montenegro, Julien Laganier, and Christoph Neumann. Hindering eavesdropping via IPv6 opportunistic encryption. In *ESORICS*, volume 3193 of *Lecture Notes in Computer Science*, pages 309–321. Springer, 2004.

[19] Claude Castelluccia and Pars Mutaf. Hash-based dynamic source routing. In *IFIP Networking*, volume 3042 of *Lecture Notes in Computer Science*, pages 1012–1023. Springer, 2004.

[20] Claude Castelluccia and Pars Mutaf. Shake them up ! : a movement-based pairing protocol for cpu-constrained devices. In *Proceedings of the 3rd ACM conference on Mobile systems, applications, and services*, pages 51–64, New York, NY, USA, 2005. ACM Press.

[21] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Efficient aggregation of encryption data in wireless sensor networks. In *IEEE MOBIQUITOUS*, 2005.

[22] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Improving secure server performance by re-balancing ssl/tls handshakes. In *ASIACCS '06 : Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 26–34, New York, NY, USA, 2006. ACM Press.

[23] Claude Castelluccia and Mate Soos. Secret shuffling : A novel approach to rfid private identification. In *Conference on RFID Security (RFIDSec07)*, 2007.

[24] C.Castelluccia, A. Chan, E.Meykletun, and G.Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Transaction on Sensor Networks (ToSN)*, 2008.

[25] C.Castelluccia, P.Mutaf, and G.Tsudik. Mouvement-based secure pairing protocols for cpu-contrained. *Submitted to ToSN (Transactions on Sensor Networks)*, 2008.

[26] Aldar Chan and Claude Castelluccia. On the security of concealed data aggregation. In *European Symposium On Research in Computer Security (ESORICS 2007)*, Dresden, Germany, September 2007.

[27] Aldar Chan and Claude Castelluccia. On the (im)possibility of aggregate message authentication code. In *IEEE International Symposium on Information Theory (ISIT 2008)*, Ontario, Canada, July 2008.

[28] Castelluccia Claude, Saxena Nitesh, and Yi Jeong. Self-configurable key pre-distribution in mobile ad-hoc network. In *IFIP Networking 2005*, 2005.

[29] Castelluccia Claude, Saxena Nitesh, and Yi Jeong. Robust self-keying mobile ad hoc networks. *Elsevier Computer Networks*, 51(4), 2007.

[30] T. Ernst, L. Bellier, and C.Castelluccia. *Mobile Networks Support in Mobile IPv6*, July 2000. IETF Internet Draft.

[31] T. Ernst, C. Castelluccia, and H. Lach. Extending mobile IPv6 with multicast to support mobile networks in IPv6. In *1st European Conference on Universal Multiservice Networks (ECUMN)*, 1999.

[32] T. Ernst, C. Castelluccia, and H. Lach. Les réseaux mobiles dans IPv6. In *13ème congrès DNAC (De Nouvelles Architectures pour les Communications)*, 1999.

[33] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks, 2002.

[34] A. Francillon and C. Castelluccia. Tinyrng, a cryptographic random number generator for wireless sensor network nodes. In *5th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, IEEE WiOpt 2007*, 2007.

[35] A. Francillon and C. Castelluccia. How to 0wn a wireless sensor network in your spare time. In *submitted to publication*, 2008.

[36] J. Kempf and C.Castelluccia. *Requirements and Functional Architecture for an IP Host Alerting Protocol*, August 2001. IETF Request For Comments, RFC 3154.

[37] G. Montenegro and C. Castelluccia. *SUCV Identifiers and Addresses*. IETF Internet Draft, 2002.

[38] Gabriel Montenegro and Claude Castelluccia. Crytographically-based identifiers (CBID) : Concepts and applications. *ACM Transaction on Information and System Security (TISSEC)*, 7(1), 2004.

[39] P. Mutaf and C. Castelluccia. A hash-based paging and location update procedure. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt'03)*, 2003.

[40] P. Mutaf and C. Castelluccia. (in)security of the paging channel in a wireless internet. In *IEEE Workshop on Applications and Services in Wireless Networks*, 2003.

[41] Pars Mutaf and Claude Castelluccia. Compact neighbor discovery : a bandwidth defense through bandwidth optimization. In *INFOCOM*, pages 2711–2719. IEEE, 2005.

[42] Pars Mutaf and Claude Castelluccia. Hash-based paging and location update using bloom filters. *ACM/Kluwer Journal on Mobile Networks and Applications (MONET)*, 10(2), 2005.

[43] H. Soliman, C.Castelluccia, K. Elmalky, and L.Bellier. *Hierarchical Mobile IPv6 Mobility Management (HMIPv6)*, August 2005. IETF Request For Comments, RFC 4140.

[44] Mate Soos and Claude Castelluccia. Noisy secret shuffling. In *Submitted to Publication*, 2008.

[45] S.Peter, D.Westhoff, and C.Castelluccia. A survey on the encryption of convergecast-traffic with in-network processing. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2008.

[46] Xinhua Zhao, Claude Castelluccia, and Mary Baker. Flexible network support for mobility. In *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 1998.

[47] Xinhua Zhao, Claude Castelluccia, and Mary Baker. Flexible network support for mobile hosts. *Mobile Networks and Applications*, 6(2) :137–149, 2001.

Annexe A

Sécurité des réseaux ad-hoc mobiles (MANET)

Cette annexe contient un article qui a été publié dans la revue *Elsevier Computer Networks, Volume 51, Issue 4, March 2007, Pages 1169-1182, March 2007.*

Cet article présente deux protocoles d'échange de clés qui sont complétement décentralisés et autonomes. Ces protocoles sont basés sur des algorithmes de partage de secret à seuil. Nous évaluons, dans cet article, leur performance et leur sécurité respective.

Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 1169–1182
Computer Networks
www.elsevier.com/locate/comnet

Robust self-keying mobile ad hoc networks ☆

Claude Castelluccia [a,1], Nitesh Saxena [b], Jeong Hyun Yi [c,*,1]

[a] *INRIA, France*
[b] *Computer Science Department, University of California, Irvine, United States*
[c] *Networking Technology Lab, Samsung Advanced Institute of Technology, Republic of Korea*

Received 30 April 2006; received in revised form 26 June 2006; accepted 12 July 2006
Available online 15 August 2006

Responsible Editor: X.S. Shen

Abstract

Pairwise key establishment in mobile ad hoc networks allows any pair of nodes to agree upon a shared key. This is an important security service needed to secure routing protocols, and in general to facilitate secure communication among the nodes of the network.

We present two self-keying mechanisms for pairwise key establishment in mobile ad hoc networks which do not require any centralized support. The mechanisms are built using the well-known technique of threshold secret sharing, and are robust and secure against a collusion of up to a certain number of nodes. We evaluate and compare the performance of both the mechanisms in terms of the node admission and pairwise key establishment.

Keywords: Mobile ad hoc networks; Security; Key management

1. Introduction

Mobile ad hoc networks (MANETs) are, by their very nature, vulnerable to many types of attacks. The security of MANETs is often predicated on the availability of efficient key management techniques. However, the usual features of: (1) lack of a centralized authority and (2) dynamic nature of MANETs, represent major obstacles to providing secure, effective and efficient key management. What further complicates the issue is that, in many applications (such as secure routing [9,8,21]) cryptographic keys need to be established *prior* to communication. As a result, standard key exchange solutions, e.g., Station-to-Station protocol [17], are not appropriate since: (1) they require the nodes to interact and (2) they rely on some form of a Public Key Infrastructure (PKI) which is not usually available in MANETs. Related to the latter is the underlying use of public key cryptography which is too expensive for some mobile devices.

☆ A preliminary version of this paper appeared in [6].

* Corresponding author. Tel.: +82 10 2481 0826.
E-mail addresses: claude.castelluccia@inrialpes.fr (C. Castelluccia), nitesh@ics.uci.edu (N. Saxena), jeong.yi@samsung.com (J.H. Yi).

[1] This work has been done while at UC Irvine, United States.

1389-1286/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2006.07.009

Contributions: This paper proposes two efficient, fully distributed and secure key management mechanisms for MANETs. The so called *self-keying* mechanisms allow nodes in a MANET to establish pairwise keys without communicating and without the need of a PKI. The first mechanism, called *matrix based self-keying* (*MSK*), results from the blending of two well-known techniques: Blom's key pre-distribution [2,13] and threshold secret sharing [25], and the second mechanism, referred to as *polynomial based self-keying* (*PSK*), employs threshold secret sharing using a polynomial. In both *MSK* and *PSK*, a node joins a MANET by receiving a secret token from t different nodes, where t is a security parameter. The schemes are *auto-configurable* in the sense that there is no centralized support required and a node becomes a member only if it is approved by at least t member nodes. Once a node becomes member, it can compute a secret key with any other member without interaction. The proposed schemes are secure against collusion of up to a certain number $(t-1)$ of compromised nodes.

The contribution of this paper is not limited to just the design of secure and efficient key distribution schemes. We also demonstrate our claims of efficiency via extensive analysis and experiments. The schemes have been implemented and tested in a real MANET setting and their performance is compared and analyzed in detail.

Organization: The rest of this paper is organized as follows: Section 2 overviews the related work. Section 3 provides some background on necessary cryptographic building blocks. Sections 5 and 6 present our self-keying mechanisms *MSK* and *PSK* respectively. We discuss some security and other relevant issues of the proposed schemes in Section 7. Finally, in Section 8, we describe the implementation and the performance of our schemes.

2. Related work

Key distribution can be easily achieved if we assume the existence of a PKI. However, this assumption is not realistic in many MANET environments. Zhou and Haas [26] proposed to distribute a Certification Authority (CA) service among several nodes of the network. Although attractive, this idea is not applicable to MANETs. Their approach is hierarchical: only selected nodes can serve as part of the certification authority and thus take part in admission decisions. Moreover, contacting the distributed CA nodes in a MANET setting is difficult since such nodes might be many hops away.

In a related result, Kong et al. [12] developed an interesting Threshold-RSA (TS-RSA) scheme specifically geared for MANETs. Unfortunately, as pointed out in [19,11], TS-RSA is neither verifiable nor secure. An alternative Threshold-DSA (TS-DSA) scheme [19] provides verifiability and, hence, tolerates malicious insiders. However, TS-DSA requires $2t-1$ signers to issue certificates, is heavily interactive and thus become quite inefficient in MANET settings. Moreover, all these solutions require a pair of nodes to perform key exchange protocol to establish shared keys.

Recently, Zhu et al. [27] proposed a pairwise key distribution scheme based on the combination of probabilistic key sharing and threshold secret sharing. However, it is assumed that the nodes are *pre-configured* with some secrets before deployment which is not realistic in a typical MANET environment. Furthermore, two nodes need to communicate over several distinct paths to establish a shared key. In contrast, we do not assume any such pre-configuration and do not require nodes to communicate when establishing a secret key.

Ćapkun et al. proposed a security association establishment protocol that makes use of the mobility of users [5]. Two nodes establish a security association when they are near each other, by using secure channels. As a node moves around, it establishes more and more security associations. When a node needs to establish a secret with another node, there are two possibilities: (1) they already have a security association, or (2) they have no security association and must use the help of "friends" to establish one. Despite the simplicity and elegance of this approach, it is mainly geared for highly mobile MANETs. Furthermore, key derivation among two nodes that do not have a prior security association requires some communication, which is not always practical or even possible.

More closely related results [7,14] present key pre-distribution schemes based on the schemes by Blom [2] and Blundo et al. [3], respectively. These schemes, unlike the one we propose in this paper, are designed for sensor networks and require a trusted centralized authority for key distribution. Similarly, the trivial solution that consists of configuring each node with pairwise keys, i.e., where a node stores $n-1$ keys, one each it shares with every other node, is not appropriate in MANETs, since (1) this requires a

centralized trusted party to compute and distribute the pairwise keys and (2) every time a new node joins the network, all the current nodes need to be updated. This is clearly not acceptable in a dynamic and volatile environment, like a MANET.

3. Building blocks

This section describes the main techniques used in our proposal, namely threshold secret sharing and Blom key pre-distribution schemes.

Following is the notation used in the rest of this paper:

M_i	member i.e., network node i
t	node admission threshold
λ	number of private keys that M_i must store
N	maximum size of network nodes
NID	network identity
id_i	crypto-based identifier of M_i
K_{ij}	secret key shared between M_i and M_j
$r_i(A)$	row of matrix A for M_i
$ss_i(x)$	secret share of value x for M_i
$pss_j^i(x)$	partial secret share of x for M_i by M_j
SL_i	sponsor list for M_i to reconstruct a secret
$H(x)$	hash value on input x
$E_k(x)$	encryption with a key k on input x
$MAC(k,x)$	message authentication code with key k on input x

3.1. Threshold secret sharing

A (t,n) threshold cryptography allows n parties to share the ability to perform a cryptographic operation in a way that any t parties can perform this operation jointly, whereas no coalition of up to $t-1$ parties can do so. We use Shamir's secret sharing scheme [25] which is based on polynomial interpolation. To distribute shares among n users, a trusted dealer TD chooses a large prime q, and selects a polynomial $f(x) = S + a_1x + \cdots + a_{t-1}x^{t-1}$ over $\mathbb{Z}_q$ of degree $t-1$ such that $f(0) = S$, where S is the group secret. The TD computes each user's share ss_i such that $ss_i = f(id_i) \pmod q$, and securely transfers ss_i to user M_i. Then, any group of t members who have their shares can recover the secret using the Lagrange interpolation formula: $f(0) = \sum_{i=1}^{t} ss_i\, l_i(0) \pmod q$, where $l_i(x) = \prod_{j=1, j\neq i}^{t} \frac{x-id_j}{id_i-id_j} \pmod q$. To enable the verification of the secret shares, TD publishes a commitment to the polynomial as in *Verifiable Secret Sharing* (VSS) [24]. VSS setup involves a large prime p such that q divides $p-1$ and a generator g which is an element of $\mathbb{Z}_p^*$ of order q. TD computes W_i $(i = 0,\ldots,t-1)$, called the *witness*, such that $W_i = g^{a_i} \pmod p$ and publishes these W_i's in some public domain (e.g., a directory server). On receiving the secret share ss_i from M_i, M_j verifies the correctness of ss_i by checking $g^{ss_i} = \prod_{k=0}^{t-1}(W_k)^{id_i^k} \pmod p$.

3.2. Blom's key pre-distribution

Blom proposed a key pre-distribution scheme that allows any pair of users in a group to compute a pairwise key without communicating [2]. This scheme is secure unless λ users collude (the parameter λ will be defined later). If less than λ users collude, then it is proven that the system is completely secure i.e., the colluding nodes cannot compute any pairwise keys other than their own. However, if λ or more users collude, the whole group is compromised and the colluding users can compute the pairwise keys of all other members.

In Blom's proposal, a trusted dealer TD computes a $\lambda \times N$ matrix B over $\mathbb{Z}_q$, where N is the maximum size of the group, q is a prime, and $q > N$.

One example of such a matrix is a Vandermonde matrix whose element $b_{ij} = (g^j)^i \pmod q$ as seen below, where g is the primitive element of $\mathbb{Z}_q^*$.

$$B = \left[b_{ij} = (g^j)^i \pmod q\right] \text{for } i,j = 1,\ldots,\lambda.$$

Note that this construction requires that $N\lambda < \phi(q)$ i.e., $N\lambda < q-1$.

Since B is a Vandermonde matrix, it can be shown that any λ columns are linearly independent when $g, g^2, g^3, \ldots, g^N$ are all distinct [15]. The TD then creates a random $\lambda \times \lambda$ symmetric matrix D over $\mathbb{Z}_q$, and computes an $N \times \lambda$ matrix $A = (DB)^{\mathrm{T}}$, where T indicates a transposition of the matrix.

The matrix B is published while the matrix D is kept secret by the TD. Since D is symmetric, the *key matrix* $K = AB$ is also symmetric

$$K = (DB)^{\mathrm{T}}B = B^{\mathrm{T}}D^{\mathrm{T}}B = B^{\mathrm{T}}DB = (AB)^{\mathrm{T}} = K^{\mathrm{T}}.$$

This shows that K is also a symmetric matrix.

We assume that each user, M_i, is defined by an identifier, id_i, such that $0 < i < N$. The TD then sends, over a secret channel, to each user M_i, the

ith row of the matrix A, denoted as $r_i(A)$, i.e., $r_i(A) = [a_{ij}]$ for $j = 1, \ldots, \lambda$.

A user M_i can then compute its key with user M_j as follows: $[K_{ij} = \sum_{\beta=1}^{\lambda} a_{i\beta} \cdot b_{\beta j}]$, where $b_{\beta j}$ is the element of B at row β and column j.

This key can be computed without communication since $b_{\beta j} = (g^j)^{\beta} \pmod q$. Similarly, user M_j can then compute its key with user M_i as follows: $[K_{ji} = \sum_{\beta=1}^{\lambda} a_{j\beta} \cdot b_{\beta i}]$. Since K is symmetric we have $K_{ij} = K_{ji}$, i.e., users M_i and M_j share a secret key. Note that each node does not have to store the whole matrix B only if he knows the public parameter g.

Since each pairwise key is represented by an element in $\mathbb{Z}_q$, q must be selected as the smallest prime number larger than 2^l, where l is the size in bits of the pairwise keys, for example 64.

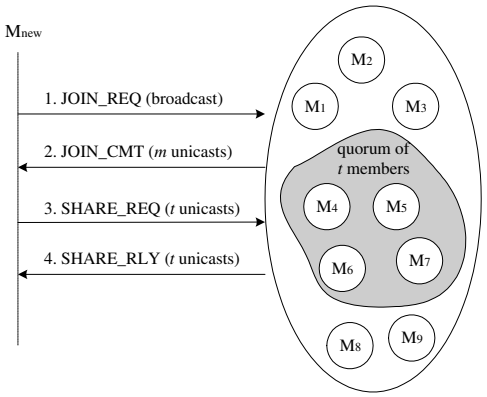


Fig. 1. Abstract Admission Protocol.

4. Generic self-keying mechanism

A self-keying mechanism for mobile ad hoc networks consists of various steps. We summarize these steps for a generic mechanism as follows:

1. *Bootstrapping*: The network is bootstrapped by either one single founding member or a set of founding members. The founding member(s) initialize the network by computing the private and corresponding public parameters. The private parameters are secret shared among the founding member(s) in such a way that any set of t members can reconstruct these parameters. The share of the private parameters possessed by each member is referred to as its secret credential.
2. *Member admission*: A prospective member M_{new} who wishes to join the network must be issued its secret credential by the existing member nodes (see Fig. 1). M_{new} initiates the admission protocol by sending a `JOIN_REQ` message to the network. A member node, that receives this `JOIN_REQ` message and approves the admission of M_{new}, replies, over a secure channel (refer to Section 7), with a partial secret credential (derived from its secret credential) for M_{new}. Once M_{new} receives partial secret credentials from at least t different nodes, it uses them to compute it secret credential.
3. *Robustness via verifiability and traceability*: A malicious node can easily launch a denial-of-service (DoS) attack toward a candidate node by inserting incorrect secret shares. This attack would actually deny or disrupt the service to legitimate nodes. To deal with this important problem a node must be able to verify the validity of its reconstructed secret credential before using them. This is what we call *verifiability* in the rest of the paper.Also, when the node detects that its secret credential is not valid, it must be able to trace the bogus shares in order to replace them and/or revoke the malicious participants. This functionality is provided by the *traceability* procedures. Note that verifying the shares' origin, for example via signatures, is not enough to provide traceability since it does not protect against compromised nodes that would signed correctly but send bogus shares. Instead, *traceability* must allow to verify the validity of the shares themselves.Note that *verifiability* is always required. *Traceability* is only necessary when a node detects (from the verifiability service) that its reconstructed secrets are not valid.
4. *Secret key computation*: Each node can use its secret credential and/or the public parameters of the network to compute pairwise keys with other nodes. This allows nodes to securely communicate with other.

5. *MSK*: matrix based self-keying

In this section we describe the *MSK* scheme, which is based on Blom's key pre-distribution described in Section 3.2.

5.1. Bootstrapping

In *MSK* scheme, a network can be bootstrapped (i.e., initialized) by one node (centralized

bootstrapping) or a set of t or more nodes (distributed bootstrapping).

Centralized bootstrapping. The centralized bootstrapping proceeds as follows. First, a founding member FM generates the network parameters, namely N, λ, t, q, p, g, and the matrices $D = [d_{ij}]$ and $B = [b_{ij}]$, where N is the maximum numbers of nodes in the network, (λ, q, p, g) are the security parameters, B is $\lambda \times N$ public matrix such that $b_{ij} = (g^j)^i \pmod q$ for $i,j \in [1,\lambda]$, and D is $\lambda \times \lambda$ symmetric matrix of secrets.

Next, the FM publishes $(N, \lambda, t, q, p, g, B)$ in some public directory, but keeps D secret. It then computes the matrix A such that $A = (DB)^{\mathrm{T}}$ and sends a share of the whole matrix A to each node.

To compute the share $ss_v(D)$ for member M_v, FM selects polynomials for each element d_{ij} of $\lambda \times \lambda$ matrix D. Each polynomial is defined as follows; $f_{d_{ij}}(x) = \sum_{\alpha=0}^{t-1} \delta_{ij}^{(\alpha)} \cdot x^{\alpha} \pmod q$ such that $\delta_{ij}^{(0)} = d_{ij}$. The share of matrix D is made up of shares of its elements $ss_v(d_{ij})$. In other words, $ss_v(D) = [ss_v(d_{ij})] = [f_{d_{ij}}(v)]$ for $i,j = 1,\ldots,\lambda$.

As for $r_v(A)$ such that $r_v(A) = [a_{vj}]$ for $j = 1,\ldots,\lambda$, each element of $r_v(A)$ is simply computed by FM since it knows the secret matrix D. That is, $a_{vj} = \sum_{\beta=1}^{\lambda} d_{j\beta} \cdot b_{\beta v} \pmod q$. Then FM distributes $ss_v(D)$ and $r_v(A)$ to each M_v.

In addition, FM computes VSS witness (which will be used in the traceability procedures defined in Section 5.3.2), $W_{ij}^{(\alpha)}$, as follows: $W_{ij}^{(\alpha)} = g^{\delta_{ij}^{(\alpha)}} \pmod p$ for $i,j \in [1,\lambda]$, $\alpha \in [0, t-1]$.

Distributed bootstrapping. The network can alternatively be bootstrapped by a set of t founding members. The secret matrix D can be generated in fully distributed manner. Note that in the centralized mode, single FM is similar to a trusted third party and is, therefore, a single point of failure. In this proposal, a group of members (the founding members in our scenario) collectively compute shares corresponding to Shamir secret sharing of a random value without a centralized trusted dealer. This procedure is so-called *Joint Secret Sharing* (*JSS*) [22]. The main idea here is that the polynomials for each element d_{ij} of matrix D are constructed among t founding members themselves such that

$$f_{d_{ij}}(x) = f_{d_{ij}[1]}(x) + f_{d_{ij}[2]}(x) + \cdots + f_{d_{ij}[t]}(x),$$

where $f_{d_{ij}[k]}(x)$ is the polynomial of each founding member FM_k over $\mathbb{Z}_q$ for $k = 1,\ldots,t$.

The detailed procedures are as follows. It is assumed that all FM's of the network have previously agreed on the system parameters $(N, \lambda, t, q, p, g, B)$. To compute $ss_v(D)$, each FM_k chooses at random a polynomial $f_{d_{ij}[k]}(x) \in \mathbb{Z}_q$ of degree $(t-1)$ such that $f_{d_{ij}[k]}(x) = \sum_{\alpha=0}^{t-1} \delta_{ij[k]}^{(\alpha)} x^{\alpha} \pmod q$ for $k, v = 1,\ldots,t$, where $\delta_{ij[k]}^{(\alpha)}$ is a random secret that FM_k selects. Then, FM_k computes FM_v's share $\hat{ss}_v^{(k)}(d_{ij}) = f_{d_{ij}[k]}(v)$ for FM_v ($v \in [1,t]$), and *securely* sends it to FM_v (in particular FM_k keeps $\hat{ss}_k^{(k)}$). Note that the share values should be transmitted over the secure channel. Upon receiving $\hat{ss}_v^{(k)}(d_{ij})$, FM_v computes its share $ss_v(d_{ij})$ of the secret d_{ij} as the sum of all shares received: $ss_v(d_{ij}) = \sum_{\alpha=1}^{t} \hat{ss}_v^{(\alpha)}(d_{ij})$.

Next, as for $r_v(A) = [a_{vj}]$ for $j = 1,\ldots,\lambda$, each FM_k computes $a_{vj}^{(k)}$ for FM_v such that $a_{vj}^{(k)} = \sum_{\beta=1}^{\lambda} ss_k(d_{j\beta}) \cdot l_k(0) \cdot b_{\beta v}$ and securely sends it to FM_v. Then, each FM_v gets its own a_{vj} by summing up all a_{vj}^k's, since $a_{vj} = \sum_{k=1}^{t} a_{vj}^k = \sum_{k=1}^{t} \sum_{\beta=1}^{\lambda} ss_k(d_{j\beta}) \cdot l_k(0) \cdot b_{\beta v} = \sum_{\beta=1}^{\lambda} d_{j\beta} \cdot b_{\beta v}$.

Finally, FM_k computes VSS witness $W_{ij[k]}^{(\alpha)}$ of its own polynomial $f_{d_{ij}[k]}(x)$ such that $W_{ij[k]}^{(\alpha)} = g^{\delta_{ij[k]}^{(\alpha)}} \pmod p$ and send it to each FM_v. Then, FM_v obtains the witness $W_{ij}^{(\alpha)}$ of $f_{d_{ij}}(x)$ as follows: $W_{ij}^{(\alpha)} = \prod_{k=1}^{t} W_{ij[k]}^{(\alpha)} \pmod p$. We note that this is actually combined with the procedure for computing $ss_v(D)$ as above.

5.2. Member admission

In order to join the network, a prospective node M_η must collect at least t shares of matrix A's row η from the current member nodes and a valid share of the whole matrix D. Fig. 2 shows the protocol message flow for the member admission process.[2]

1. M_η sends to at least t current member nodes M_v's ($v \in [1,n]$) a signed `JOIN_REQ` message which contains his identity id_η and his public Diffie-Hellman (DH) component y_η $(= g^{x_\eta} \pmod p)$. The details about how id_η is generated and verified are discussed in Section 7.
2. After verifying the signed `JOIN_REQ`, the member nodes who wish to participate in the admission process of M_η reply with a signed message containing their respective values id_v and y_v.
3. M_η selects t sponsors $M_\mu (\mu \in_R v, |\mu| = t)$, computes a secret key $DHK_{\eta\mu}$ with each of them, forms a sponsor list SL_η which contains the id's of the t

[2] In order to secure the protocol against common *replay* attacks [17], we note that it is necessary to include timestamps, nonces and protocol message identifiers. However, in order to keep our description simple, we omit these values.

$$
\begin{array}{lll}
msg1(M_\eta \rightarrow M_\nu): & REQ = \{id_\eta, y_\eta\},\ S_\eta(REQ) & (1)\\
msg2(M_\eta \leftarrow M_\nu): & REP = \{id_\nu, y_\nu\},\ S_\nu(REP, H(REQ)) & (2)\\
msg3(M_\eta \rightarrow M_\mu): & SL_\eta, MAC(DHK_{\eta\mu},\ H(SL_\eta, msg1, msg2)) & (3)\\
msg4(M_\eta \leftarrow M_\mu): & E_{DHK_{\eta\mu}}\{pss_\mu(r_\eta(D)), ss_\mu(r_\eta(A))\} & (4)
\end{array}
$$

Fig. 2. *MSK* Admission Protocol.

selected sponsors, and replies with an authenticated acknowledgment message to each of them.

4. Each sponsoring node (M_μ) on receiving *msg*3, computes the secret key $DHK_{\eta\mu}$ and replies with row η of it share of the matrix A, $ss_\mu(r_\eta(A))$. The elements of $ss_\mu(r_\eta(A))$ are computed as $ss_\mu(r_\eta(a_{\eta j})) = \sum_{\beta=1}^{\lambda} ss_\mu(d_{j\beta}) \cdot b_{\beta\eta} \pmod q$, for $j = 1, \ldots, \lambda$. This message is encrypted with $DHK_{\eta\mu}$. Each (M_μ) also responds with the *shuffled* partial share of matrix D, $pss_\mu^\eta(D)$, such that $pss_\mu^\eta(D) = [pss_\mu^\eta(d_{ij})] = [ss_\mu(d_{ij}) \cdot l_\mu(id_\eta)] \pmod q$ for $i,j = 1, \ldots, \lambda$. This message is also encrypted using $DHK_{\eta\mu}$. Note that the Lagrange coefficients $l_\mu(id_\eta)$ are publicly known, and therefore, M_η can derive $ss_\mu(d_{ij})$ from $pss_\mu^\eta(d_{ij})$. This can be prevented using the *shuffling* technique proposed in [12] by adding extra random value R_{ij} to each share. These R_{ij}'s are secret values and must sum up to zero by construction. They must be securely shared among the t sponsoring nodes.
5. M_η decrypts the messages it receives from the different nodes and calculates his own $r_\eta(A)$ by adding up all $ss_\mu(r_\eta(A))$'s as follows: $r_\eta(A) = \sum_{\mu=1}^{t} ss_\mu(r_\eta(A)) \cdot l_\mu(0) = [\sum_{\mu=1}^{t} ss_\mu(r_\eta(a_{\eta j})) \cdot l_\mu(0)] \pmod q$ for $j = 1, \ldots, \lambda$. M_η also calculates his own share of the matrix D, $ss_\eta(D)$, by adding up the partial share values such that $ss_\eta(D) = \sum_{\mu=1}^{t} pss_\mu^\eta(D) = [\sum_{\mu=1}^{t} pss_\mu^\eta(d_{ij})] \pmod q$ for $i,j = 1, \ldots, \lambda$.

5.3. Robustness via verifiability and traceability

At the end of the admission protocol, the joining node M_η obtains its $r_\eta(A)$ and $ss_\eta(D)$ from the quorum of t member nodes. Before using $r_\eta(A)$ and $ss_\eta(D)$ for key computation or future admission, the node must verify if they are correctly computed since there might be a malicious responder who participated in this admission process and detect them in the process. We therefore propose following *verifiability* and *traceability* mechanisms.

5.3.1. Verifiability

The VSS technique presented in Section 3.1 will be a useful tool for the verifiability. However, we claim that the verifiability must be a very inexpensive operation since it will be performed frequently (whenever a node joins a network). The proposed mechanism to verify the validity of $r_\eta(A)$ is as follows:

1. *When an existing member node M_μ sends the shares to the node M_η it also sends a well-known message, such as "Welcome to network NID" encrypted with the pairwise key shared between M_μ and M_η (since the M_μ knows the node identifier id_η, it can compute the pairwise key $K_{\mu\eta}$). This will be part of step (4) in Fig. 2.*
2. *After M_η reconstructs its systems secrets $r_\eta(A)$, it can then try to decrypt one of the welcome messages received from the member nodes and verify whether $r_\eta(A)$ is correctly computed.*

Additionally, M_η must verify the validity of the reconstructed secret share $ss_\eta(D)$. If $ss_\eta(D)$ is correct, it can be used for future admission of other nodes. One easy way for M_η to verify the validity of $ss_\eta(D)$ is to try to use it to reconstruct its row of the matrix A (i.e., $r_\eta(A)$) as follows:

> *Let us say that $r_\eta(A)$* was computed from the shares $ss_a(r_\eta(A))$, $ss_b(r_\eta(A))$, $ss_c(r_\eta(A))$ *that it received from M_a, M_b and M_c (for $t = 3$). M_η can then compute $r'_\eta(A)$ from the shares $ss_a(r_\eta(A))$,* $ss_b(r_\eta(A))$ and $ss_\eta(r_\eta(A))$ *(that can easily be computed from $ss_\eta(D)$). If $r'_\eta(A)$ is equal to $r_\eta(A)$ then the share $ss_\eta(D)$ is correct – otherwise it must be rejected.*

5.3.2. Traceability

The verifiability procedures previously described allow a node to verify the validity of the secret (which is the row of the matrix A and a share of the whole matrix D) that it reconstructed from t shares. However, the above procedure cannot be used to identify the bogus shares, in case the verification procedure fails (i.e., detects that the reconstructed secrets are invalid).

In this section, we present two different traceability procedures. The first – *external attack traceability* – traces external malicious nodes, i.e., malicious

nodes that are not part of the MANET and just try to attack the network by sending bogus shares to new member. The second – *internal attack traceability* – is useful to detect attacks coming from current legitimate member nodes that either turn malicious or get compromised.

Both procedures use the previously described VSS technique in some innovative ways. Since the internal traceability procedure is quite costly, we recommend to use the external attack traceability first and use the internal attack traceability procedure only if the first one turns out to be unsuccessful.

External attack traceability

With this procedure, a node M_η, instead of verifying individual element of a share (row or matrix), verifies the sum of the elements of the share. As a result, instead of applying the VSS technique λ or λ^2 times, we only apply it once. This, of course, improves performance considerably.

More specifically, M_η verifies the validity of the share $ss_\mu(r_\eta(A))$ and $pss^\eta_\mu(D)$ using the VSS technique defined in Section 5.1, as follows:

M_η first computes $\sigma_{ss_\mu(r_\eta(A))}$ by summing up all elements of $ss_\mu(r_\eta(A))$; i.e., $\sigma_{ss_\mu(r_\eta(A))} = \sum_{i=1}^{\lambda} ss_\mu(a_{\eta i}) \pmod q$. Since $W_C^{(\alpha)} = \prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}(W_{ij}^{(\alpha)})^{b_{j\eta}}$ is *pre-computable*, the validity of $\sigma_{ss_\mu(r_\eta(A))}$ can be verified by checking the following equality:

$$g^{\sigma_{ss_\mu(r_\eta(A))}} \stackrel{?}{=} \prod_{\alpha=0}^{t-1}[W_C^{(\alpha)}]^{id_\mu^\alpha} \pmod p,$$

$$\text{where } W_C^{(\alpha)} = \prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}(W_{ij}^{(\alpha)})^{b_{j\eta}} \pmod p.$$

Proof. Since $ss_\mu(a_{\eta i}) = \sum_{j=1}^{\lambda} ss_\mu(d_{ij}) \cdot b_{j\eta}$, $ss_\mu(d_{ij}) = f_{d_{ij}}(id_\mu) = \sum_{\alpha=0}^{t-1}\delta_{ij}^{(\alpha)} \cdot id_\mu^\alpha \pmod q$, and $W_C^{(\alpha)} = \prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}(W_{ij}^{(\alpha)})^{b_{j\eta}}$ for $\alpha = 0, \ldots, t-1$,

$$\begin{aligned}
g^{\sigma_{ss_\mu(r_\eta(A))}} &= \prod_{\alpha=0}^{t-1}[W_C^{(\alpha)}]^{id_\mu^\alpha} \\
&= \prod_{\alpha=0}^{t-1}\left[\prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}(W_{ij}^{(\alpha)})^{b_{j\eta}}\right]^{id_\mu^\alpha} \\
&= \prod_{\alpha=0}^{t-1}\left[\prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}(g^{\delta_{ij}^{(\alpha)}})^{b_{j\eta}}\right]^{id_\mu^\alpha} \\
&= g^{\sum_{\alpha=0}^{t-1}\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda}(\delta_{ij}^{(\alpha)}\cdot b_{j\eta})\cdot id_\mu^\alpha} \\
&= g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda}\left(\sum_{\alpha=0}^{t-1}\delta_{ij}^{(\alpha)}\cdot id_\mu^\alpha\right)\cdot b_{j\eta}} \\
&= g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} ss_\mu(d_{ij})\cdot b_{j\eta}} \\
&= g^{\sum_{i=1}^{\lambda} ss_\mu(a_{\eta i})} \pmod p. \qquad \square
\end{aligned}$$

Similarly, given the precomputed $W_D^{(\alpha)} = \prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}W_{ij}^{(\alpha)}$, a node can verify the validity of a partial of the matrix D, $pss^\eta_\mu(D)$, after computing $\sigma_{pss^\eta_\mu(D)} = \sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} pss^\eta_\mu(d_{ij}) \pmod q$, as follows:

$$g^{\sigma_{pss^\eta_\mu(D)}} \stackrel{?}{=} \prod_{\alpha=0}^{t-1}[W_D^{(\alpha)}]^{id_\mu^\alpha \cdot l_\mu(id_\eta)} \pmod p,$$

$$\text{where } W_D^{(\alpha)} = \prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}W_{ij}^{(\alpha)} \pmod p.$$

Proof. Since $pss^\eta_\mu(d_{ij}) = ss_\mu(d_{ij}) \cdot l_\mu(id_\eta)$, $ss_\mu(d_{ij}) = f_{d_{ij}}(id_\mu) = \sum_{\alpha=0}^{t-1}\delta_{ij}^{(\alpha)} \cdot id_\mu^\alpha \pmod q$, and $W_D^{(\alpha)} = \prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}W_{ij}^{(\alpha)}$ for $\alpha = 0, \ldots, t-1$,

$$\begin{aligned}
g^{\sigma_{pss^\eta_\mu(D)}} &= \prod_{\alpha=0}^{t-1}[W_D^{(\alpha)}]^{id_\mu^\alpha \cdot l_\mu(id_\eta)} \\
&= \prod_{\alpha=0}^{t-1}\left[\prod_{i=1}^{\lambda}\prod_{j=1}^{\lambda}W_{ij}^{(\alpha)}\right]^{id_\mu^\alpha \cdot l_\mu(id_\eta)} \\
&= \left[g^{\sum_{\alpha=0}^{t-1}\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda}\delta_{ij}^{(\alpha)}}\right]^{id_\mu^\alpha \cdot l_\mu(id_\eta)} \\
&= g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda}\left(\sum_{\alpha=0}^{t-1}\delta_{ij}^{(\alpha)}\cdot id_\mu^\alpha\right)\cdot l_\mu(id_\eta)} \\
&= g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} ss_\mu(d_{ij}) l_\mu(id_\eta)} \\
&= g^{\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} pss^\eta_\mu(d_{ij})} \pmod p. \qquad \square
\end{aligned}$$

If the above verification fails, M_η concludes that M_μ is not a legitimate member node. Otherwise, the malicious node is a group member and thus the following procedure must be used.

Internal attack traceability

If a malicious insider (M_m), who has valid $ss_m(D)$, modifies a value in $ss_m(D)$ so that the sum of $ss_m(D)$ remains unchanged (say, $ss'_m(D)$ such that $\sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} ss'_m(d_{ij}) = \sum_{i=1}^{\lambda}\sum_{j=1}^{\lambda} ss_m(d_{ij})$), the *external attack traceability* procedure does not work. In order

to protect against such an internal attack, VSS must be applied individually to each of elements of $pss_\mu(r_\eta(A))$ and/or $ss^\eta_\mu(D)$. Since $W_I^{(\alpha)} = \prod_{i=1}^{\lambda} (W_{ij}^{(\alpha)})^{b_{j\eta}}$ is pre-computable, *internal traceability* is provided by checking that

$$g^{ss_\mu(a_{\eta i})} = \prod_{\alpha=0}^{t-1} [W_I^{(\alpha)}]^{id_\mu^\alpha} \quad \text{and}$$

$$g^{pss^\eta_\mu(d_{ij})} = \prod_{\alpha=0}^{t-1} [W_{ij}^{(\alpha)}]^{id_\mu^\alpha l_\mu(id_\eta)} \pmod p$$

$$\text{where } W_I^{(\alpha)} = \prod_{i=1}^{\lambda} (W_{ij}^{(\alpha)})^{b_{j\eta}} \pmod p, \quad i,j \in [1,\lambda].$$

Obviously, these tracing mechanisms for an internal attack are more expensive than the external ones. However, we argue that tracing is an infrequent phenomenon, as an attacker knows that if it performs an internal attack, it will be detected in the process.

5.4. Secret key computation

When a node, M_i, reconstructs its private row of matrix A, $r_i(A) = [a_{i1}, \ldots, a_{i\lambda}]$, he can compute a secret key, $K_{i\,j}$, with any other node, M_j, of the network as follows:

Since $a_{ij} = \sum_{\alpha=1}^{\lambda} d_{j\alpha} \cdot b_{\alpha i}$ and $d_{ij} = d_{ji}$,

$$K_{ij} = \sum_{\beta=1}^{\lambda} a_{i\beta} b_{\beta j} = \sum_{\beta=1}^{\lambda} \sum_{\alpha=1}^{\lambda} d_{\beta\alpha} b_{\alpha i} b_{\beta j}$$

$$= \sum_{\alpha=1}^{\lambda} \sum_{\beta=1}^{\lambda} d_{\alpha\beta} b_{\beta j} b_{\alpha i} = \sum_{\alpha=1}^{\lambda} a_{j\alpha} b_{\alpha i} = K_{ji}.$$

Note that these keys do not have to be computed in advance but can be computed *on-the-fly*. The security of this key establishment procedure is unconditional, i.e, it is not based on any security assumption. Refer to [3] for the security arguments.

6. *PSK*: polynomial based self-keying

The various steps of the *PSK* scheme, which is based on polynomial secret sharing, are described in following subsections.

6.1. Bootstrapping

Centralized bootstrapping. The centralized bootstrapping works exactly as described in Section 3.1.

Distributed bootstrapping. A group of t or more founding members employ JSS [22] to collectively compute shares corresponding to Shamir secret sharing of a random value.

6.2. Member admission

In order to join the network, a prospective node M_η must collect at least t partial shares from existing nodes to be able to compute its secret share. Fig. 3 shows the protocol message flow for the member admission process.

1–3. Steps 1–3 are exactly the same as in the *MSK* admission protocol described in Section 5.2.
4. Each sponsoring node (M_μ) on receiving *msg*3, computes the secret key $DHK_{\eta\mu}$ and replies with the *shuffled* partial share [12], $pss_\mu(\eta)$, such that $pss_\mu(\eta) = ss_\mu \cdot l_\mu(id_\eta) \pmod q$. This message is encrypted using $DHK_{\eta\mu}$.
5. M_η decrypts the messages it receives from the different nodes and calculates his own secret share ss_η, by adding up the partial share values such that $ss_\eta = \sum_{\mu=1}^{t} pss_\mu(\eta)$.

6.3. Robustness via verifiability and traceability

M_η can easily validate the acquired secret share by checking if $g^{ss_\eta} = \prod_{k=0}^{t-1} (W_k)^{id_\eta^k} \pmod p$ from the public commitment values. In case this verification fails, M_η can trace the node(s) which sent the fake shares by checking the validity of each of $pss_\mu(\eta)$ values. This can achieved by verifying if $g^{pss_\mu(\eta)} = [\prod_{k=0}^{t-1} (W_k)^{id_\mu^k}]^{l_\mu(id_\eta)} \pmod p$.

6.4. Secret key computation

Any pair of nodes M_i and M_j can establish shared keys using their respective secret shares ss_i, ss_j and the public VSS information as described in Section 3.1. M_i computes $g^{ss_j} = \prod_{k=0}^{t-1} (W_k)^{id_j^k} \pmod p$ from the public commitment values, and exponentiate it to its own share ss_i to get a key $K_{ij} = (g^{ss_j})^{ss_i} \pmod p$. Similarly, M_j computes $g^{ss_i} = \prod_{k=0}^{t-1} (W_k)^{id_i^k} \pmod p$ and exponentiate it to its own share ss_j to get a key $K_{ji} = (g^{ss_i})^{ss_j} \pmod p$. Since, $K_{ij} = K_{ji} = K$, M_i and M_j have a shared secret and they can use $H(K)$ as a symmetric key (where $H()$ is a hash function such as MD5 or SHA-1) to secure their subsequent communication.

Note that the above key establishment mechanism is different from standard Diffie-Hellman key exchange protocol. In the latter, the secret exponents used by the parties are *independently* generated, while in the former, the secret exponents (or the secret shares) are *related* by being points on a polynomial and any set of t of these exponents

$$
\begin{array}{lll}
msg1(M_\eta \rightarrow M_\nu): & REQ = \{id_\eta, y_\eta\},\ S_\eta(REQ) & (1)\\
msg2(M_\eta \leftarrow M_\nu): & REP = \{id_\nu, y_\nu\},\ S_\nu(REP, H(REQ)) & (2)\\
msg3(M_\eta \rightarrow M_\mu): & SL_\eta, MAC(DHK_{\eta\mu},\ H(SL_\eta, msg1, msg2)) & (3)\\
msg4(M_\eta \leftarrow M_\mu): & E_{DHK_{\eta\mu}}\{pss_\mu(\eta)\} & (4)
\end{array}
$$

Fig. 3. *PSK* Admission Protocol.

determine the rest. It is not obvious whether such a usage of related exponents in the key establishment remains secure.

We show, in the theorem to follow, that indeed the proposed key establishment procedure using related exponents remains secure. Basically, we show that our scheme remains secure under the Computational Diffie-Hellman (CDH) assumption[3] in the random oracle model (ROM) [1]. In other words, an adversary who corrupts atmost $t-1$ nodes, cannot distinguish a key K_{IJ} for some uncorrupted user pair (M_I, M_J) from random *even* if he learns all other session keys K_{ij} for $(i,j) \neq (I,J)$, as long as the CDH assumption holds and when hash function is modeled as an ideal random oracle. This is the standard notion for the security of a key establishment protocol and is adopted from [4].

Theorem 1 (security of PSK secret key computation). *Under the CDH Assumption in ROM, there exists no probabilistic polynomial time adversary A, which on inputs of secret keys of t corrupted users, and shared keys K_{ij} between every user pair except K_{IJ} $\{(i,j) \neq (I,J)\}$, is able to distinguish with a non-negligible probability K_{IJ} from a random value.*

Proof. We prove the above claim by contradiction, i.e, we prove that if a polynomial time adversarial algorithm A, which on inputs of secret keys of t corrupted users, and shared keys K_{ij} between every user pair except K_{IJ} $\{(i,j) \neq (I,J)\}$, is able to distinguish with a non-negligible probability K_{IJ} from a random value, then there exists a polynomial time algorithm B, which is able to break the CDH assumption in the random oracle model.

In order to construct the algorithm B which breaks the CDH assumption, we first construct a polynomial time algorithm C, which breaks the Square Computational Diffie-Hellman (SCDH)[4] assumption. The algorithm C runs on input of an SCDH instance $y = g^x \pmod p$, and would translate the adversarial algorithm A into outputting $g^{x^2} \pmod p$.

Without loss of generality, we first assume that the adversary A corrupts $t-1$ players denoted by $M_1, M_2, \ldots, M_{t-1}$. Now, the algorithm C runs as follows:

As in the simulation of Feldman's VSS, C picks $x_1, x_2, \ldots, x_{t-1}$ values corresponding to the secret keys of corrupted users, uniformly at random from $\mathbb{Z}_q$. It then sets $x_i = F(id_i)$, and employs appropriate Lagrange interpolation coefficients in the exponent to compute the public witnesses $g^{A_1}, \ldots, g^{A_{t-1}} \pmod p$, where $F(z) = x + A_1 z + \cdots + A_{t-1} z^{t-1} \pmod q$.

Corresponding to the shared keys K_{ij} between every user pair, C picks a random value R_{ij}, and runs the algorithm A on $x_1, \ldots, x_{t-1}$ and $R_{i,j}$ values. Note that the values $x_1, \ldots, x_{t-1}$ and the witnesses have an identical distribution to an actual run of the Feldman's secret sharing protocol, and therefore A cannot see the difference between C's inputs and actual protocol run. Also, since the K_{ij} values for $(i,j) \neq (I,J)$ are obtained by hashing $g^{x_i x_j}$, the only way A can tell the difference, except with negligible probability, between $K_{i,j}$ and $R_{i,j}$ for $(i,j) \neq (I,J)$, is by querying the random oracle on at least one appropriate $g^{x_i x_j}$ value. If A does tell the difference, then C records $R = g^{x_i x_j}$, and use the following equations to compute g^{x^2},

$$x = \sum_{k=1}^{t-1} x_k l_k^i + x_i l_i^i \pmod q,$$

$$x = \sum_{k=1}^{t-1} x_k l_k^j + x_j l_j^j \pmod q$$

(l_k^i denotes the Lagrange coefficient $l_k^G(0)$, where $G = \{1, \ldots, t-1, i\}$).

Multiplying above two equations, we get

$$x^2 = \left(\sum_{k=1}^{t-1} x_k l_k^i\right)\left(\sum_{k=1}^{t-1} x_k l_k^j\right) + x_i x_j l_i^i l_j^j \pmod q.$$

[3] CDH assumption: In a cyclic group generated by $g \in Z_p^*$ of order q, for $a, b \in \mathbb{Z}_q^*$, given $(g, g^a \pmod p, g^b \pmod p)$, it is hard to compute $g^{ab} \pmod p$.

[4] SCDH assumption: In a cyclic group generated by $g \in Z_p^*$ of order q, for $a \in \mathbb{Z}_q^*$, given $(g, g^a \pmod p)$, it is hard to compute $g^{a^2} \pmod p$.

This implies,

$$g^{x^2} = g^{\left(\sum_{k=1}^{t-1} x_k I_k^i\right)\left(\sum_{k=1}^{t-1} x_k I_k^j\right)} R^{I_i^i I_j^j} \pmod{p}.$$

If A does not tell the difference between $K_{i,j}$ and $R_{i,j}$ for $(i,j) \neq (I,J)$, then it must tell the difference between $K_{I,J}$ and $R_{I,J}$. However, as above, this is only possible, except with negligible probability, if A queries $g^{x_I x_J}$ to the random oracle. Them C records this value (say K) and computes g^{x^2} similarly as above, using the following equation:

$$g^{x^2} = g^{\left(\sum_{k=1}^{t-1} x_k I_k^i\right)\left(\sum_{k=1}^{t-1} x_k I_k^j\right)} K^{I_i^i I_j^j} \pmod{p}.$$

Now, we will use C to construct B to break a CDH instance (g^u, g^v). This is very simple as outlined in [16]: B runs C on input g^u, then on g^v, and finally on $g^{u+v} = g^u g^v$, and receives g^{u^2}, g^{v^2}, $g^{(u+v)^2}$, respectively. Now, since $(u+v)^2 = u^2 + v^2 + 2uv \pmod{q}$, B can easily compute g^{uv} from the outputs of C.

Clearly, $Pr(B) = Pr(C)^3$, where $Pr(B)$, $Pr(C)$, denote the probabilities of success of B and C respectively. □

7. Discussion

7.1. Identifier configuration

In the *MSK* and *PSK* schemes, the identifier id_i of each node M_i must be *unique* and *verifiable*. Otherwise, a malicious node could use the identifier of some other node and get its secret from the member nodes during the admission process.

For unique and unforgeable *id* assignment, we propose to use a solution based on *Crypto-Based ID* (*CBID*) [18]: The id_i is chosen by the node itself from an ephemeral public/private key pair. More specifically, the node computes id_i as follows: $id_i = H_{64}(y_i|NID)$, where y_i is M_i's temporary DH public key in our schemes, *NID* is the network identifier and $H_{64}(\cdot)$ a 64-bit long hash function. When a node contacts the member nodes for admission, it sends its identifier id_i together with its ephemeral public key y_i and signs a challenge sent by the member node. Upon reception of the signature, the member node can verify that the id_i actually belongs to the requesting node (by verifying the signature and that the id_i) was generated as $H_{64}(y_i|NID)$. Note that the y_i does not need to be certified and therefore no PKI is required. The identifier is *verifiable* because a node that does not know the private key, associated with the public key used to generate an *id*, cannot claim to own it. Furthermore since *id* is computed from a hash function, collision probability between two nodes is very low. As a result, the identifier are *statistically unique*. Note that this solution requires that $N = 2^{64}$. However, as we will see this has no effect on the performance or scalability of our proposal.

7.2. Secure channel establishment

In the proposed admission protocols, the channels between the node requesting admission and each of the member nodes must be authenticated and encrypted. It has to be authenticated because each member node must be sure that it is sending the shares to the correct node (i.e., the node that claims to own the identifier). Otherwise, the member node could send the shares to an impersonating node. Similarly, the joining node also needs to authenticate the member nodes. The channel has to be private because otherwise a malicious node that eavesdrops on the shares sent to a node could reconstruct the node's secret and impersonate it.

Establishing an authenticated and private channel usually requires the use of certificates, which bind identities to public keys, and an access to a PKI. However, PKI is not always available in MANET environments. Fortunately in our case, what is really needed is a way to bind an identifier to a public key, where the identifier is a number that identifies one row of the matrix A. This binding is actually provided by *CBID*, described previously. As a result, certificates and PKI are not required. Therefore, the public keys (y_i's) that are sent in message 1 and 2 of the protocols described in Sections 5.2 and 6.2 do not need to be certified.

7.3. Parameters selection

The security of the *MSK* scheme relies on two security parameters t and λ, whereas the *PSK* scheme depends only on t. λ and t denote the number of collusions needed to break these schemes. These parameters should be selected carefully. In particular, it is suggested to set $\lambda = t$. However, more generally, λ should be at least t in the hierarchical MANET settings where only a subset of nodes possesses the ability to admit new nodes. For the evaluation of our schemes (as described in the next section) we set $\lambda \geqslant t$.

8. Performance evaluation

We implemented both *MSK* and *PSK* protocols and evaluated them in a real MANET environment in terms of node admission and pairwise key computation costs.

8.1. Experimental setup

The *MSK* and *PSK* protocol suites are implemented on top of the OpenSSL library [20]. They are written in C for Linux, and consists of about 10,000 lines of code for each. The source code is available at [23].

For the experimental setup, we used a total of five laptops; four laptops with a Pentium-3 800 MHz CPU and 256MB memory and one laptop with a Mobile Pentium 1.8 GHz CPU and 512MB memory. Each device ran Linux 2.4 and was equipped with a 802.11*b* wireless card configured in ad-hoc mode. Specifically, for measuring the admission cost, four laptops with same computing power were used to configure the existing member nodes and the high-end laptop was used for the joining node. In our experiments, each node (except the joining node) was emulated by a daemon and each machine was running up to three daemons. The measurements were performed with the different threshold values t and λ for *MSK*. The size of the parameters q was set to 160 bits and p to 512 or 1024 bits.

8.2. Admission cost

To evaluate the admission cost, we measured the total processing time between the sending of the `JOIN_REQ` by the prospective node and the receiving (plus verification) of acquired credentials (i.e., $r_\eta(A)$ and $ss_\eta(D)$ in *MSK* and ss_η in *PSK*). The resulting measurements include the average computation time of the basic operations, the communication costs such as packet encoding and decoding time, the network delay, and so on.

Fig. 4 shows the average admission time for the joining node for different values of the threshold t. For the *MSK* testing, λ was set to 3, 5, 7 and 9. (In the figure, λ is denoted by L.)

As observed from the graphs, the cost for a node to join the network with *PSK* is cheaper than that of *MSK*. This difference in the costs between *MSK* and *PSK* is even higher for higher threshold values. The reason is quite intuitive: *MSK* requires more computation and bandwidth than *PSK*. More specifically, the *MSK* scheme requires $O(\lambda^2 t)$ *multiplications* and $O(\lambda^2)$ *exponentiations* whereas *PSK* requires only $O(t^2)$ *multiplications* and O(1) *exponentiations*. For the bandwidth costs, refer to Table 1.

This table shows that the *PSK* scheme is very efficient in terms of bandwidth. This is an important property for MANET systems which consist of battery-operated devices, because wireless transmission is considered as the most energy consuming operation.[5]

8.3. Traceability cost

As described in Sections 5.3 and 6.3, the traceability procedures are used to identify the cheating or misbehaving nodes during the admission protocols. This section evaluates the performance of these procedures.

Fig. 5 displays the cost of both the internal and external attack traceability procedures. As for the external attack traceability, denoted as *MSK_EXT*, the cost is slightly expensive than *PSK* since some expensive operations are pre-computable in the latter. In details, the computation complexity for both *MSK_EXT* and *PSK* is the same; i.e., $O(t)$. The cost of the internal traceability procedure with *MSK*, denoted as *MSK_INT*, depends on the value of λ as well as t. As a result, this cost increases when λ gets larger. The complexity of the *MSK* internal attack traceability procedure is $O(\lambda^2 t)$ exponentiations with modulus p. However, we expect these procedures to be executed very infrequently only when the external traceability fails.

8.4. Key computation cost

Table 2 compares the cost of computing a pairwise key in our schemes. The results show that *MSK* performs significantly better than a *PSK* protocol. The achieved gains with $\lambda = 9$ range from 10 ($t = 1$) to 13 ($t = 9$), and from 305 to 307 for 512-bit and 1024-bit p, respectively. In other words, *MSK* is 10–307 times faster than *PSK* when establishing a shared secret key.

These results were actually expected because in *MSK* the pairwise computation requires only $O(\lambda)$ modular *multiplications* where the modulus size is 160 bits. In contrast, *PSK* requires $O(t)$ expensive

[5] It has been shown that sending one bit of data is roughly equivalent to adding 1000 32-bit numbers [19].

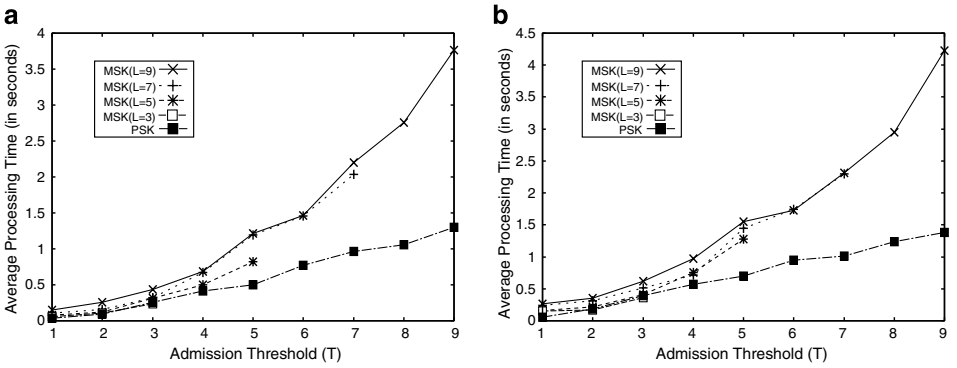


Fig. 4. Admission cost (a): $|p| = 512$ and (b) $|p| = 1024$.

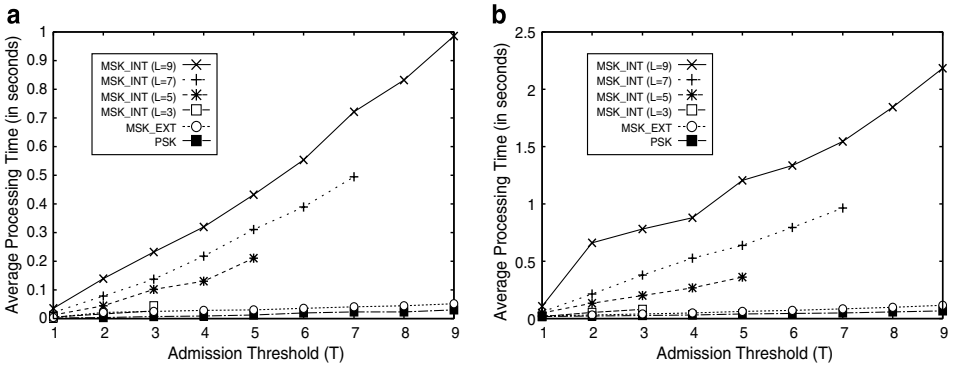


Fig. 5. Traceability cost: (a) $|p| = 512$ and (b) $|p| = 1024$.

Table 1
Bandwidth comparison

	MSK	*PSK*
Admission	$O(\lambda t\|q\|) + O(\lambda^2\|q\|)$	$O(t\|q\|)$
Shuffling	$O(\lambda^2 t^2\|q\|)$	$O(t^2\|q\|)$

Table 2
Key computation cost (in ms, P4-3.0 GHz, 1 GB memory)

t	*MSK* ($\lambda \geqslant t$)				*PSK*	
	$\lambda = 3$	$\lambda = 5$	$\lambda = 7$	$\lambda = 9$	$\|p\| = 512$	$\|p\| = 1024$
1	0.0371	0.0301	0.0430	0.0550	0.574	17.780
2	0.0398	0.0415	0.0506	0.0570	0.683	18.150
3	0.0436	0.0424	0.0568	0.0564	0.713	18.180
4	–	0.0365	0.0595	0.0655	0.663	18.220
5	–	0.0431	0.0565	0.0629	0.753	18.370
6	–	–	0.0628	0.0563	0.772	18.450
7	–	–	0.0562	0.0629	0.782	18.570
8	–	–	–	0.0644	0.851	18.540
9	–	–	–	0.0637	0.871	19.120

modular *exponentiations* with a modulus size of 512 or 1024 bits.

9. Conclusion

We presented distributed solutions to the key predistribution problem in MANETs. Our self-keying solutions, *MSK* and *PSK*, are based on the secret sharing techniques and are secure against collusive attacks by a certain threshold of nodes. The solutions allow any pair of nodes in the network to establish shared keys *without communication*, as opposed to the standard Diffie-Hellman key exchange protocols. We implemented the *MSK* and *PSK* schemes and evaluated them in real MANET setting. Our analysis show that *MSK* fares better than *PSK* as far as the pairwise key establishment costs are concerned. However, in terms of the node admission costs, the latter outperforms the former. Based on

this analysis, we conclude that the *MSK* scheme is well-suited for MANET applications where node admission is not a frequent operation, whereas the *PSK* scheme is more applicable for highly dynamic MANETs consisting of mobile devices with reasonably high computation power.

References

[1] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: ACM Conference on Computer and Communications Security, 1993, pp. 62–73.

[2] R. Blom, An optimal class of symmetric key generation systems, in: EUROCRYPT'84, LNCS, IACR, 1984.

[3] C. Blundo, A.D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, perfectly-secure key distribution for dynamic conferences, in: Advances in Cryptology – CRYPTO'92, 1992, pp. 471–486.

[4] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in: EUROCRYPT'01, Springer-Verlag, 2001, pp. 453–474.

[5] S. Capkun, J.-P. Hubaux, L. Buttyan, Mobility helps security in ad hoc networks, in: 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'03), 2003, pp. 46–56.

[6] C. Castelluccia, N. Saxena, J.H. Yi, Self-configurable key pre-distribution in mobile ad hoc networks, in: IFIP Networking Conference, May 2005, pp. 1083–1095.

[7] W. Du, J. Deng, Y.S. Han, P.K. Varshney, A pairwise key pre-distribution scheme for wireless sensor networks, in: Jajodia et al. [10], pp. 42–51.

[8] Y.-C. Hu, D.B. Johnson, A. Perrig, Sead: secure efficient distance vector routing in mobile wireless ad hoc networks, in: Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02), June 2002, pp. 3–13.

[9] Y.-C. Hu, A. Perrig, D.B. Johnson, Ariadne: a secure on-demand routing protocol for ad hoc networks, in: Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking, MobiCom 2002.

[10] S. Jajodia, V. Atluri, T. Jaeger (Eds.), Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, 27–30 October 2003, ACM, 2003.

[11] S. Jarecki, N. Saxena, J.H. Yi, An attack on the proactive RSA signature scheme in the URSA ad hoc network access control protocol, in: ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), October 2004, pp. 1–9.

[12] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, Providing robust and ubiquitous security support for MANET, in: IEEE 9th International Conference on Network Protocols (ICNP), 2001.

[13] T. Leighton, S. Micali, Secret-key agreement without public-key cryptography, in: CRYPTO'93, 1993.

[14] D. Liu, P. Ning, Establishing pairwise keys in distributed sensor networks, in Jajodia et al. [10], pp. 52–61.

[15] F.J. MacWilliams, N. Sloane, The Theory of Error-Correcting Codes, North Holland, Amsterdam, 1997.

[16] U.M. Maurer, S. Wolf, Diffie-Hellman oracles, in: CRYPTO'96, LNCS, vol. 1109, 1996, pp. 268–282.

[17] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press Series on Discrete Mathematics and Its Applications, 1997, ISBN 0-8493-8523-7.

[18] G. Montenegro, C. Castelluccia, Crypto-based identifiers (cbids): concepts and applications, ACM TISSEC 7 (1) (2004).

[19] M. Narasimha, G. Tsudik, J.H. Yi, On the utility of distributed cryptography in P2P and MANETs: the case of membership control, in: IEEE International Conference on Network Protocol (ICNP), November 2003, pp. 336–345.

[20] OpenSSL Project, http://www.openssl.org/.

[21] P. Papadimitratos, Z. Haas, Secure Routing for Mobile Ad Hoc Networks, 2002.

[22] T.P. Pedersen, A threshold cryptosystem without a trusted party, in: D. Davies (Ed.), EUROCRYPT'91, IACR, 1991, LNCS, vol. 547, pp. 552–526.

[23] Peer Group Admission Control Project. Available from: <http://sconce.ics.uci.edu/gac>.

[24] P. Feldman, A practical scheme for non-interactive verifiable secret sharing, in: 28th Symposium on Foundations of Computer Science (FOCS), 1987, pp. 427–437.

[25] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.

[26] L. Zhou, Z.J. Haas, Securing ad hoc networks, IEEE Network Mag. 13 (6) (1999) 24–30.

[27] S. Zhu, S. Xu, S. Setia, S. Jajodia, Establishing pair-wise keys for secure communication in ad hoc networks: a probabilistic approach, in: IEEE International Conference on Network Protocol (ICNP), November 2003.

Claude Castelluccia is a senior research scientist at INRIA, France. He received an engineering degree from the Universite de Technologie de Compiegne (90), France, a Master of Science in EE from Florida Atlantic University (92) and a Ph.D. in Computer Science from INRIA (96). He was a post-doctoral fellow at Stanford University in 1997 and a visiting researcher at University of California Irvine from 2003 to 2005. His research interests are in network security, applied cryptography, mobile/wireless networking. wireless sensor networks and RFID.

Nitesh Saxena is an assistant professor in the department of Computer and Information Science at Polytechnic University, starting Fall 2006. He obtained his Ph.D. in Information and Computer Science from University of California, Irvine, in summer 2006. He holds an M.S. degree in Computer Science from UC Santa Barbara, and a Bachelor's degree in Mathematics and Computing from Indian Institute of Technology, Kharagpur, India. Nitesh's research spans all areas of information security with core emphasis on network and distributed system security and applied cryptography. Nitesh's Ph.D. dissertation entitled ''Decentralized Security Services'' has been nominated for the ACM Dissertation Award for the year 2006.

Jeong Hyun Yi is a principal researcher in Samsung Advanced Institute of Technology (SAIT). He received his Ph.D. in Information and Computer Science with his advisor, Dr. Gene Tsudik, from University of California, Irvine in 2005. He received his M.S. and B.S. in Computer Science at Soongsil University, Korea in 1995 and 1993, respectively. He was a senior researcher at Electronics and Telecommunication Research Institute (ETRI), Korea from 1995 to 2001 and a guest researcher at National Institute of Standards and Technology (NIST), MD, USA from 2000 to 2001. His research interests are in network security, applied cryptography, ubiquitous computing, RFID and wireless sensor networks.

Annexe B

Agrégation de données chiffrées

Cette annexe contient un article qui a été accepté pour publication dans la revue *ACM Transactions on Sensor Networks (ToSN)*. Une version préliminaire a également été publiée dans la conférence ACM Mobiquitous (International Conference on Mobile and Ubiquitous Systems : Computing, Networking and Services) en 2005.

Cet article présente un nouveau algorithme de chiffrement homomorphique par l'addition, c'est à dire qui permet l'addition de données chiffrées. Il présente également un protocole d'agrégation des données chiffrées pour les réseaux de capteurs. Ce protocole est très peu coûteux en terme de calcul et est, par conséquent, très bien adapté aux réseaux de capteurs. L'article présente également une preuve de la sécurité de ce protocole.

Efficient and Provably Secure Aggregation of Encrypted Data in Wireless Sensor Networks

CLAUDE CASTELLUCCIA
INRIA
ALDAR C-F. CHAN
National University of Singapore
EINAR MYKLETUN, GENE TSUDIK
University of California, Irvine

Wireless sensor networks (WSNs) are multi-hop networks composed of tiny devices with limited computation and energy capacities. For such devices, data transmission is a very energy-consuming operation. It thus becomes essential to the lifetime of a WSN to minimize the number of bits sent by each device. One well-known approach is to aggregate sensor data (e.g., by adding) along the path from sensors to the sink. Aggregation becomes especially challenging if end-to-end privacy between sensors and the sink or aggregate integrity is required. In this paper, we propose a simple and provably secure encryption scheme that allows efficient (additive) aggregation of encrypted data. Only one modular addition is necessary for ciphertext aggregation. The security of the scheme is based on the indistinguishability property of a pseudorandom function, a standard cryptographic primitive. We show that aggregation based on this scheme can be used to efficiently compute statistical values such as mean, variance and standard deviation of sensed data, while achieving significant bandwidth gain. To protect the integrity of data incorporated in an aggregate, we also give an end-to-end aggregate authentication scheme which is secure against outsider-only attacks based on the indistinguishability property of a pseudorandom function.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Sensor Networks*

General Terms: Security, Design

Additional Key Words and Phrases: Wireless Sensor Networks, Secure Data Aggregation, Stream Ciphers, Privacy, Authentication

Some of our preliminary results were originally published in [Castelluccia et al. 2005]. The present paper represents a reworked and extended version of [Castelluccia et al. 2005]. Major new components include the security analysis and the technique for authentication of encrypted aggregated data.
Part of this work was done while Claude Castelluccia was visiting at the University of California, Irvine

Author's address:
Claude Castelluccia, INRIA, Zirst - 655 avenue de l'Europe, 38334 Saint Ismier Cedex, France. Email: Claude.Castelluccia@inria.fr
Aldar C-F. Chan, Department of Computer Science, School of Computing, National University of Singapore, Singapore 117543. Email: aldar@graduate.hku.hk
Einar Mykletun, Gene Tsudik, Computer Science Department, School of Information and Computer Science, University of California, Irvine. Email: {mykletun, gts}@ics.uci.edu

1. INTRODUCTION

Wireless sensor networks (WSNs) are becoming increasingly popular in many spheres of life. Application domains include monitoring of the environment (such as temperature, humidity and seismic activity) as well as numerous other ecological, law enforcement and military settings.

Regardless of the application, most WSNs have two notable properties in common: (1) the network's overall goal is typically to reach a collective conclusion regarding the outside environment, which requires detection and coordination at the sensor level, and (2) WSNs act under severe technological constraints: individual sensors have severely limited computation, communication and power (battery) resources and need to operate in settings with great spatial and temporal variability

At the same time, WSNs are often deployed in public or otherwise untrusted and even hostile environments, which prompts a number of security issues. These include the usual topics, e.g., key management, privacy, access control, authentication and DoS-resistance, among others. What exacerbates and distinguishes security issues in WSNs is the need to miniaturize all security services so as to minimize security-induced overhead. In other words, if security is a necessary hindrance in other (e.g., wired or MANET) types of networks, it is much more so in WSNs. For example, public key cryptography is typically ruled out[1] as are relatively heavy-weight conventional encryption methods.

Security in WSNs is a popular research topic and many advances have been reported on in recent years. Most prior work has focused on ultra-efficient key management, authentication, routing and DoS resistance [Eschenauer and Gligor 2000; Zhu et al. 2004; Karlof and Wagner 2003; Wood and Stankovic 2002]. An overview of security related issues and services required for WSNs is provided by Perrig, et al. in [Perrig et al. 2004].

On the other hand, a lot of attention has been devoted to communication efficiency issues. Since data transmission is a very energy-consuming operation, in order to maximize sensor lifetime, it is essential to minimize the sheer number of bits sent by each sensor device. One natural and well-known approach involves aggregating sensor data as it propagates along the path from the sensors to the so-called sink – a node that collects sensed data. Of course, aggregating data is not quite equivalent to collecting individual sensor readings. In some applications, e.g., perimeter control, aggregation is not applicable since only individual sensor readings are of interest. However, many WSN scenarios that monitor an entire micro-environment (e.g., temperature or seismic activity) do not require information from individual sensors but, instead, put more emphasis on statistical quantities, such as mean, median and variance.

End-to-end privacy and aggregate integrity/authenticity are the two major security goals of a secure WSN. Regardless of information leakage due to the correlation among sensor measurements, end-to-end privacy ensures that nobody other than the sink could learn considerable information about the final aggregate even if he might control any subset of sensor nodes. Informally speaking, aggregate authentication provides assurance that the final aggregate does not deviate too far away from what can be achieved at the sink when all sensor nodes act honestly and nobody has tampered the data en route, or the deviation will be detected at the sink.

Although simple and well-understood, aggregation becomes problematic if end-to-end privacy between sensors and the sink is required. If we assume that all sensors are trusted, sensors could encrypt data on a hop-by-hop basis. For an intermediate sensor (i.e., one that receives

[1]While many sensor devices could have sufficient computation power to perform operations in public key cryptography, transmitting a ciphertext (from public-key cryptography) could be overwhelming to most sensor devices. Note that typical ciphertext size in a practical public-key cryptosystem is about 1024 bits or more.

and forwards data), this would entail: 1) sharing a key with each neighboring sensor, 2) decrypting encrypted messages received from each child, 3) aggregating all received values, and 4) encrypting the result for transmission to its father. Though viable, this approach is fairly expensive and complicated, the former because of having to decrypt each received value before aggregation and the latter due to the overhead imposed by key management. Furthermore, hop-by-hop encryption assumes that all sensors are trusted with the authenticity and privacy of other sensors' data. This assumption may be altogether unrealistic in some setting, whereas, in others, trust can be partial, i.e., intermediate nodes are trusted with only authenticity or only privacy.

Alternatively, if a single global key was used by all sensors, by subverting a single sensor node the adversary could learn measured values of any and all nodes in the network. Since only the sink should gain an overview of WSN measurements, this approach is not attractive. Nevertheless, we do not rule out using a single global key for message authentication (of the aggregate), which is another challenging security goal in WSNs. In fact, aggregate authentication against outsider-only attacks might be the best one can achieve for end-to-end integrity in the WSN scenario. In other words, additive aggregate authentication secure against malicious insiders might not be achievable without using leverage like outlier detection or range checking. These techniques have to be based on a stronger assumption that the statistical distribution of measurements is known (partially or completely) beforehand; note that they are essentially data/aggregate plausibility checks [Wagner 2004]. When an attacker can inject a contribution of arbitrary value into an additive aggregate through compromised insiders, he can actually manipulate the final aggregate by any amount of deviation he likes without being detected.

Contributions: In this paper, we focus on efficient, bandwidth-conserving privacy in WSNs. More specifically, we blend inexpensive encryption techniques with simple aggregation methods to achieve very efficient aggregation of encrypted data. To assess the practicality of proposed techniques, we evaluate them and present very encouraging results which clearly demonstrate appreciable bandwidth conservation and small overhead stemming from both encryption and aggregation operations. We also provide a security argument of the proposed encryption scheme. More specifically, we prove that the proposed scheme achieves semantic security if the encryption keys are generated by a good pseudorandom function family. With a view to supporting aggregate integrity, we also extend the proposed encryption scheme to provide end-to-end aggregate authentication which is provably secure against outsider-only attacks.

Organization: In the next section we discuss some background and the assumptions about our system model. Then, Section 3 describes the problem statement along with the security model. Next, Section 4 describes our homomorphic encryption scheme, followed by Section 5 which describes how to utilize this encryption scheme in a WSN. We give a security proof of the proposed scheme in Section 6. Performance is analyzed and results are discussed in Section 7. The aggregate authentication scheme and its security analysis is given in Section 8. Related work is summarized in Section 9 and Section 10 concludes this paper.

2. BACKGROUND

In this section we describe the key features of, and assumptions about, the network and provide an overview of aggregation techniques.

2.1 Wireless Sensor Networks (WSNs)

A WSN is a multi-hop network composed of a multitude of tiny devices with limited computation and energy capacities. One commonly cited WSN application is monitoring the environment. This may include sensing motion, measuring temperature, humidity, etc. Data monitored by the sensors is sent to a sink (usually a more powerful device), that is responsible for collecting the information.

The multi-hop nature of a WSN implies that sensors are also used in the network infrastructure, i.e., not just sending their own data and receiving direct instructions but also forwarding data for other sensors. When sensors are deployed, a *delivery tree* is often built from the sink to all sensors. Packets sent by a sensor are forwarded to the sink by the sensors along the delivery tree.

Sensor nodes come in various shapes and forms, however, they are generally assumed to be resource-limited with respect to computation power, storage, memory and, especially, battery life. A popular example is the Berkeley mote [Madden et al. 2002]. One common sensor feature is the disproportionately high cost of transmitting information as compared to performing local computation. For example, a Berkeley mote spends approximately the same amount of energy to compute 800 instructions as it does in sending a single bit of data [Madden et al. 2002]. It thus becomes essential to reduce the number of bits forwarded by intermediate nodes, in order to extend the entire network's lifetime. The sink node acts as a bridge between the WSN and the outside world. It is typically a relatively powerful device, such as a laptop computer.

2.2 Aggregation in WSN

Aggregation techniques are used to reduce the amount of data communicated within a WSN and thus conserves battery power. Periodically, as measurements are recorded by individual sensors, they need to be collected and processed to produce data representative of the entire WSN, such as average and/or variance of the temperature or humidity within an area. One natural approach is for sensors to send their values to certain special nodes, i.e., aggregating nodes. Each aggregating node then condenses the data prior to sending it on. In terms of bandwidth and energy consumption, aggregation is beneficial as long as the aggregation process is not too CPU-intensive.

The aggregating nodes can either be special (more powerful) nodes or regular sensors nodes. In this paper, we assume that all nodes are potential aggregating nodes and that data gets aggregated as they propagate towards the sink. In this setting, since sensors have very limited capabilities, aggregation must be simple and not involve any expensive or complex computations. Ideally, it would require only a few simple arithmetic operations, such as additions or multiplications.[2].

We note that aggregation requires all sensors to send their data to the sink within the same sampling period. This either requires the sensors to have (at least loosely) synchronized clocks or the ability to respond to explicit queries issued by the sink.

One natural and common way to aggregate data is to simply add up values as they are forwarded towards the sink. Of course, this type of aggregation is useful when the sink is only interested in certain statistical measurements, e.g., the mean or variance of all measured data. As noted in Section 1, some WSN applications require all sensor data and therefore can not benefit from aggregation techniques. Similarly, applications requiring boundary values, e.g.,

[2]This is indeed what we achieve in this work.

min and/or max, are obviously not a good match for additive aggregation.

With additive aggregation, each sensor sums all values, x_i, it receives from its k children (in the sink-rooted spanning tree) and forwards the sum to its parent. Eventually, the sink obtains the sum of all values sent by all n sensors. By dividing the sum by n, i.e., the total numbers of sensors, it computes the average of all measured data.

This simple aggregation is very efficient since each aggregating node only performs k arithmetic additions[3]. It is also robust since there is no requirement for all sensors to participate as long as the sink gets the total number of sensors that actually provided a measurement.

Additive aggregation can be also used to compute the variance, standard deviation and any other moments on the measured data. For example, in case of variance, each aggregating node not only computes the sum, $S = \sum_{i=1}^{k} x_i$, of the individual values sent by its k children, but also the sum of their squares: $V = \sum_{i=1}^{k} x_i^2$. Eventually, the sink obtains two values: the sum of the actual samples which it can use to compute the mean and the sum of the squares which it can use to compute the variance:

$Var = E(x^2) - E(x)^2$; where
$E(x^2) = (\sum_{i=1}^{n} x_i^2)/n$ and $E(x) = (\sum_{i=1}^{n} x_i)/n$

3. GOALS AND SECURITY MODEL

To provide data privacy, our goal is to prevent an attacker from gaining any information about sensor data beside what can be inferred by the measurements done directly by the attacker. We define the privacy goal by the standard notion of semantic security in this work.

An attacker is assumed to be global, i.e., able to monitor any location in the network or even the entire WSN. Furthermore, we assume the attacker is able to *read* the internal state of some sensors. The attacker is also supposed to be able to corrupt a subset of sensor nodes. We assume the attacker can launch chosen plaintext attacks only. That is, the attacker is able to obtain the ciphertext of any plaintext of his choice. In the real situation, this means the attacker could manipulate the sensing environment and obtain the desired ciphertext by eavesdropping.

In light of our requirement for end-to-end privacy between the sensors and the sink, additive aggregation, although otherwise simple, becomes problematic. This is largely because popular block and stream ciphers, such as AES [NIST 2001] or RC5 [Rivest 1995], are not additively homomorphic. In other words, the summation of encrypted values does not allow for the retrieval of the sum of the plaintext values.

To minimize trust assumptions, we assume that each of the n sensors shares a distinct long-term key with the sink, called the encryption key. This key is originally derived, using a pseudo-random function (PRF), from the master secret, which is only known to the sink. We denote the sink's master secret as K and the long-term sensor/sink shared key as ek_i, where the subscript $0 < i \leq n$ uniquely identifies a particular sensor. This way, the sink only needs to store a single master secret and all long-term keys can be recomputed as needed.

As opposed to encryption, authentication schemes that allow for aggregation seem to be very difficult, and perhaps impossible, to design. It should be noted that the problem of aggregate authentication considered in this paper is different from the problem considered in aggregate signatures [Boneh et al. 2003]; more precisely, the latter should be called aggregatable signa-

[3]We assume that an aggregating node has its own measurement to contribute; thus k additions are needed.

tures instead. In aggregate authentication, it is the messages themselves being aggregated and hence the original messages are not available for verification, whereas, in aggregate signatures, the signatures for different messages are aggregated and all the signed messages have to be distinct and available to the verification algorithm in order to verify the validity of an aggregate signature. Consequently, it is fair to say there could be no secure aggregate authentication scheme (which is existentially unforgeable against chosen message attacks) in the literature. As explained in [Wagner 2004], other techniques are likely needed to verify the plausibility of the resulting aggregate and to increase the aggregation resiliency.

In WSNs, providing end-to-end aggregate authentication seems to be difficult since messages lose their entropies through aggregation, rendering it difficult to verify the validity of a given aggregate. But it is still possible to prevent unauthorized nodes from injecting fake packets in the networks. That is, groupwise message authentication can be achieved in which only nodes knowing a common group key can contribute to an aggregate and produce valid authentication tags that would pass a prescribed verification test at the sink. Note that the scheme would be vulnerable to compromised nodes. We give an end-to-end message authentication scheme providing such access control assuming outsider-only attacks.

4. ADDITIVELY AGGREGATE ENCRYPTION

Encrypted data aggregation or aggregate encryption is sometimes called concealed data aggregation (CDA), a term coined by Westhoff et. al. [Westhoff et al. 2006]. Appendix A gives an abstract description of CDA showing the desired functionalities.

In this section we describe the notion of *homomorphic encryption* and provide an example. Our notion is a generalized version of the widely used one for homomorphic encryption — we allow the homomorphism be under different keys while the homomorphism in common notions is usually under the same key. We then proceed to present our additively homomorphic encryption scheme whose security analysis is given in Section 6 and Appendix B. The encryption technique is very well-suited for privacy-preserving additive aggregation. For the sake of clarity, in Section 4.2, we will first describe a basic scheme assuming the encryption keys are randomly picked in each session (which is the same scheme as given in our earlier work [Castelluccia et al. 2005]); the header part is also excluded in the discussion. Then we will give a concrete construction in which the session keys and the encryption keys are derived using a pseudorandom function family. The concrete construction can be proved to be semantically secure in the CDA model [Chan and Castelluccia 2007], the details of which are given in Appendix A. Compared to our earlier work [Castelluccia et al. 2005], this paper provides the details of a concrete construction using a pseudorandom function in Section 4.3, with the security requirements on the used components specified.

Our scheme can be considered as a practical, tailored modification of the Vernam cipher [Vernam 1926], the well-known one-time pad, to allow plaintext addition to be done in the ciphertext domain. Basically, there are two modifications. First, the exclusive-OR operation is replaced by an addition operation. By choosing a proper modulus, multiplicative aggregation is also possible.[4] Second, instead of uniformly picking a key at random from the key space, the key is generated by a certain deterministic algorithm (with an unknown seed) such as a pseudorandom function [Goldreich 2001]; this modification is actually the same as that in a

[4]Our construction can achieve either additive or multiplicative aggregation but not both at the same time. Besides, multiplication aggregation seems to bear no advantage as the size of a multiplicative aggregate is the same as the sum of the size of its inputs.

stream cipher. As a result, the information-theoretic security (which requires the key be at least as long as the plaintext) in the Vernam cipher is replaced with a security guarantee in the computational-complexity theoretic setting in our construction.

4.1 Homomorphic Encryption

A homomorphic encryption scheme allows arithmetic operations to be performed on ciphertexts. One example is a multiplicatively homomorphic scheme, whereby the multiplication of two ciphertexts followed by a decryption operation yields the same result as, say, the multiplication of the two corresponding plaintext values. Homomorphic encryption schemes are especially useful in scenarios where someone who does not have decryption keys needs to perform arithmetic operations on a set of ciphertexts. A more formal description of homomorphic encryptions schemes is as follows.

Let $Enc()$ denote a probabilistic encryption scheme. Let M be the message space and C the ciphertext space such that M is a group under operation $\oplus$ and $Enc()$ is a $\oplus$-homomorphic encryption scheme if for any instance $Enc()$ of the encryption scheme, given $c_1 = Enc_{k_1}(m_1)$ and $c_2 = Enc_{k_2}(m_2)$ for some $m_1, m_2 \in M$, there exists an efficient algorithm which can generate from c_1 and c_2 a valid ciphertext $c_3 \in C$ for some key k_3 such that

$$c_3 = Enc_{k_3}(m_1 \oplus m_2)$$

In other words, decrypting c_3 with k_3 would yield $m_1 \oplus m_2$. In this paper, we mainly consider additive homomorphism, i.e. $\oplus$ is the $+$ operation. We do not restrict k_1, k_2, k_3 to be the same despite that they are usually equal in common homomorphic encryption schemes. Since k_3 could be different from k_1, k_2, some identifying information, say, denoted by hdr, needs to be attached to a ciphertext to indicate which keys are required to decrypt the ciphertext.

A good example is the RSA cryptosystem[Rivest et al. 1978] which is *multiplicatively homomorphic* under a single key. The RSA encryption function is $Enc(m) = m^e = c \pmod n$ and the corresponding decryption function is $Dec(c) = c^d = m \pmod n$ where n is a product of two suitably large primes (p and q), e and d are encryption and decryption exponents, respectively, such that $e * d = 1 \pmod{(p-1)(q-1)}$.

Given two RSA ciphertexts c_1 and c_2, corresponding to respective plaintexts m_1 and m_2, it is easy to see that $c_1 c_2 \equiv m_1^e m_2^e \equiv (m_1 m_2)^e \pmod n$. Hence, one can easily compute the multiplication of the ciphertexts $(c_1 c_2)$ to obtain the ciphertext corresponding to the plaintext $m = m_1 m_2 \pmod n$. Note that c_1, c_2 and the resulting ciphertext after multiplication are all under the same decryption key d and no hdr is thus needed.

4.2 Basic Encryption Scheme using Random Keys

We now introduce a simple *additively homomorphic* encryption technique. The main idea of our scheme is to replace the *xor* (Exclusive-OR) operation typically found in stream ciphers with modular addition (+). *For the sake of clarity, the inclusion of hdr (the information to identify decryption keys) and pseudorandom functions is deferred to the discussion in Section 4.3.* The basic scheme is as follows.

Basic Additively Homomorphic Encryption Scheme

Encryption:

(1) Represent message m as an integer $m \in [0, M-1]$ where M is the modulus of arithmetics.
(2) Let k be a randomly generated keystream, where $k \in [0, M-1]$.
(3) Compute $c = Enc_k(m) = m + k \bmod M$.

Decryption:

(1) $Dec_k(c) = c - k \bmod M$.

Addition of Ciphertexts:

(1) Let $c_1 = Enc_{k_1}(m_1)$ and $c_2 = Enc_{k_2}(m_2)$.
(2) The aggregated ciphertext is: $c_l = c_1 + c_2 \bmod M = Enc_k(m_1 + m_2)$ where $k = k_1 + k_2 \bmod M$.

The correctness of the aggregation is assured if M is sufficiently large. The explanation is as follows: $c_1 = m_1 + k_1 \bmod M$ and $c_2 = m_2 + k_2 \bmod M$, then $c_l = c_1 + c_2 \bmod M = (m_1+m_2)+(k_1+k_2) \bmod M = Enc_{k_1+k_2}(m_1+m_2)$. For $k = k_1+k_2$, $Dec_k(c_l) = c_l - k \bmod M = (m_1 + m_2) + (k_1 + k_2) - (k_1 + k_2) \bmod M = m_1 + m_2 \bmod M$.

We assume that $0 \leq m < M$. Note that if n different ciphers c_i are added together, then M must be larger than $\sum_{i=1}^{n} m_i$, otherwise *correctness* is not provided. In fact if $\sum_{i=1}^{n} m_i$ is larger than M, decryption will result in a value m' that is smaller than M. In practice, if $t = max_i\{m_i\}$, then M should be selected as $M = 2^{\lceil log_2(t*n) \rceil}$. That is, the required length of M should be at least the sum of the length of t and the length of n.

Note that this basic scheme is only for illustration purposes and is not the actual construction. Since the encryption key k is assumed to be randomly picked (as in the one-time pad) by the sensor node in each reporting session, a secure channel has to be maintained at all time between each sensor node and the sink. In the actual construction (given in Section 4.3), such a secure channel is not required.

4.3 Encryption Scheme using Keys generated by a Pseudorandom Function Family

The main difference between the actual construction and the basic encryption scheme is that the encryption keys in each session are now generated by a pseudorandom function instead of being randomly picked. Two components are used in the constructions, namely, a pseudorandom function f and a length-matching hash function h. Their details are as follows.

4.3.1 *Pseudorandom Functions.* A pseudorandom function is used for deriving secret keys. For details on pseudorandom functions, [Goldreich 2001] has a comprehensive description. Let $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ be a pseudorandom function family where $F_\lambda = \{f_s : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda\}_{s \in \{0,1\}^\lambda}$ is a collection of functions indexed by a key $s \in \{0,1\}^\lambda$. Loosely speaking, given a function f_s from a pseudorandom function ensemble with unknown key s, any PPT distinguishing procedure allowed to get the values of $f_s(\cdot)$ at (polynomially many) arguments of its choice should not be able to tell (with non-negligible advantage in λ) whether the answer of a new query is supplied by f_s or randomly picked from $\{0,1\}^\lambda$.

Most provably secure pseudorandom functions such as [Naor et al. 2002] are based on the hardness of some number-theoretic problems. However, these constructions are usually com-

putationally expensive for sensor nodes. Instead, key derivation in practice is usually based on functions with conjectured or assumed pseudorandomness, that is, pseudorandomness or unpredictability is inherently assumed in the construction rather than proven to follow from the hardness of some computational problems. One typical example is the use of cryptographic hash functions for key derivation such as [Perrig et al. 2001]. Even HMAC [Bellare et al. 1996] and OMAC [Iwata and Kurosawa 2003] are constructed based on assumed pseudorandomness, with the former assuming the underlying hash function in the construction has a certain pseudorandomness property and the latter assuming the block cipher in use is a pseudorandom permutation.

The proposed additive aggregate encryption scheme in this paper does not pose a restriction on which type of pseudorandom functions should be used. A conjectured pseudorandom function can be used for the sake of efficiency. The security guarantee provided by the proposed construction holds as long as the underlying pseudorandom function has the widely defined property of pseudorandomness or indistinguishability. If such an indistinguishability property no longer holds or is broken for the pseudorandom function in use, we can simply replace it with a better pseudorandom function (with its indistinguishability property yet to be broken) for the proposed aggregate encryption to remain secure. It should be emphasized that the mentioned indistinguishability or pseudorandomness property is also an inherent requirement on the hash function to be used as a key derivation function [Perrig et al. 2001; Bellare et al. 1996] which is also used in the IPSec standard. That is, if a given hash is not suitable for the proposed aggregate encryption scheme due to its weakness, the same weakness would also undermine the security foundation of these key derivation functions.

4.3.2 *Length-matching Hash Function.* The length-matching hash function $h : \{0,1\}^\lambda \rightarrow \{0,1\}^l$ matches the length of the output of the pseudorandom function f to the modulus size of M, that is, M is assumed to be l bits long. The purpose of h is to shorten a long bit-string rather than to produce a fingerprint of a message; hence, unlike cryptographic hash functions, h is not required to be collision resistant. The only requirement on h is: $\{t \leftarrow \{0,1\}^\lambda : h(t)\}$ has a uniform distribution over $\{0,1\}^l$. That is, by uniformly picking an input from the domain of h, the resulting output distribution is uniform over the range of h.

This requirement is pretty loose and many compression maps from $\{0,1\}^\lambda$ to $\{0,1\}^l$ work. For instance, h can be implemented by truncating the output of the pseudorandom function and taking the least significant l bits as output. The sufficiency of this requirement on h is based on the assumption that an ideal pseudorandom function is used. For such a function, without knowledge of the seed key, it is unpredictable whether an output bit is 0 or 1 for all input. In practice, key derivation is usually based on conjectured pseudorandom functions with unproven pseudorandomness; for example, a collision resistant hash function is commonly used for deriving secret keys from a seed [Perrig et al. 2001; Bellare et al. 1996]. Hence, it might be the case that, for some input to these conjectured pseudorandom functions, there is a higher chance (greater than $\frac{1}{2}$) to predict some output bit successfully. To tolerate the imperfectness of conjectured pseudorandom functions, if $l|\lambda$, a better construction could be as follows: truncate the output of the pseudorandom function into smaller strings of length l and then take exclusive-OR on all these strings and use it as the output of h.

Assume there is a sink and n nodes in the system. In the following description, f is a pseudorandom function for key stream generation and h is a length-matching hash function. The details of the proposed aggregate encryption scheme are as follows.

Additively Homomorphic Encryption Scheme using a Pseudorandom Function Family

Assume the modulus is M.

Key Generation:

(1) Randomly pick $K \in \{0,1\}^{\lambda}$ and set it as the decryption key for the sink.
(2) For each $i \in [1,n]$, set the encryption key for node i as $ek_i = f_K(i)$.

Encryption:

(1) Given an encryption key ek_i, a plaintext data m_i and a nonce r, output $c_i = Enc_{ek_i}(m_i) = m_i + h(f_{ek_i}(r)) \bmod M$.
(2) Set the header $hdr_i = \{i\}$.
(3) Output (hdr_i, c_i) as the ciphertext.

Decryption:

(1) Given the ciphertext (hdr, c) of an aggregate and a nonce r used in the encryption, generate $ek_i = f_K(i), \forall i \in hdr$.
(2) Compute $x = Dec_K(c) = (c - \sum_{i \in hdr} h(f_{ek_i}(r))) \bmod M$ (where $K = \sum_{i \in hdr} h(f_{ek_i}(r))$), and output the plaintext aggregate x.

Addition of Ciphertexts:

(1) Given two CDA ciphertexts (hdr_i, c_i) and (hdr_j, c_j), compute $c_l = (c_i + c_j) \bmod M$
(2) Set $hdr_l = hdr_i \cup hdr_j$.
(3) Output (hdr_l, c_l).

The keystream for a node is now generated from its secret key ek_i and a unique message ID or nonce r. No randomness in the nonce is needed. This secret key is pre-computed and shared between the node and the sink, while the nonce can either be included in the query from the sink or derived from the time period in which the node is sending its values in (assuming some form of synchronization).

5. AGGREGATION OF ENCRYPTED DATA

As previously noted, efficient aggregation in WSNs becomes very challenging when end-to-end privacy of data is required. One solution is to disregard aggregation altogether in favor of privacy, i.e., for sensor nodes to forward to their parents their own encrypted measurements, as well as measurements received from their children. The sink, upon receiving as many data packets as there are responding sensors, proceeds to decrypt all ciphertexts and sums them up in order to compute the desired statistical measurements. We term this approach as *No-Agg*. This approach has two obvious disadvantages. First, because all packets are forwarded towards the sink, a lot of bandwidth (and hence power) is consumed. Second, as illustrated later in Section 7.2, there is an extreme imbalance between sensors in terms of the amount of data communicated. Sensors closer to the sink send and receive up to several orders of magnitude more bits than those on the periphery of the spanning tree.

A second approach, that does not achieve end-to-end privacy but does aggregate data, is a *hop-by-hop (HBH)* encryption method, which is also used for comparison between aggregation methods in [Girao et al. 2004]. In HBH all nodes create pair-wise keys with their parents and children during a boot strapping phase. When answering a query, nodes decrypt any packets

sent to them, aggregate this data together with their own before re-encrypting the aggregated result and forwarding this to their parent. This approach is obviously more bandwidth efficient than No-Agg, as no packet is sent twice. However, there is an associated cost involved with the decryption and encryption performed at every non-leaf node in the WSN which increases their energy consumption (see [Girao et al. 2004]). More importantly, from a privacy perspective, the HBH scheme leaves nodes vulnerable to attacks because their aggregated data will appear in plaintext (i.e., no end-to-end privacy). Especially nodes closer to the sink become attractive targets for an attacker, as their aggregated values represent a large portion of the data in the WSN.

We instead propose an end-to-end privacy preserving aggregation approach (denoted as *AGG*) in which each sensor encrypts their sensed data using the encryption scheme presented in Section 4.3. Since this scheme is additively homomorphic, values can be added (aggregated) as they are forwarded towards the sink. The sink can then retrieve from the aggregate it receives the sum of the samples and derive certain statistical data. AGG retains the positive qualities of both the No-Agg (end-to-end privacy) and HBH (energy efficient) solutions. Note that a piece of identifying information is needed for each ciphertext in No-Agg to allow the sink to decide which key to use for decrypting a particular ciphertext in the list of received ciphertexts. This identifying information has roughly the same size as *hdr* in *AGG*.

5.1 Computing Statistical Data

In this section, we show how the new additively homomorphic encryption scheme can be used to aggregate encrypted data such that the sink can still compute the average and variance. Since multiple moduli may be used for different instances of the aggregate encryption scheme in the following discussion, the modulus used for encryption and decryption will be explicitly specified in the notation for clarity. For example, $Enc_k(x; M)$ means encrypting x using key k with public parameter M (the modulus).

5.1.1 *Computing the Average.* When using our scheme, each sensor encrypts its data x_i to obtain $c_{x_i} = Enc_{k_i}(x_i; M)$. M needs to be chosen large enough to prevent an overflow so it is set as $M = n * t$, where t is the range of possible measurement values and n is the number sensor nodes. Each ciphertext c_{x_i} is therefore $log(M) = log(t) + log(n)$ bits long.

The sensor then forwards c_{x_i} along with the key identifying information hdr_{x_i} to its parent, who aggregates all the c_{x_j}'s of its k children by simply adding them up (this addition is performed modulo M). The resulting value is then forwarded. The sink ends up with value $C_x = \sum_{i=1}^{n} c_{x_i} \bmod M$ associated with *hdr* which indicates the key set $\{k_1, ..., k_i, ..., k_n\}$. It can then compute $S_x = Dec_K(C_x; M) = C_x - K \bmod M$, where $K = \sum_{i=1}^{n} k_i$, and derive the average as follows: Avg $= S_x/n$.

5.1.2 *Computing the Variance.* As mentioned previously, our scheme can also be used to derive the variance of the measured data. Two moduli will be used, M for the sum of values and M' for the sum of squares.

In this case, each sensor i must compute $y_i = x_i^2$, where x_i is the measured sample, and encrypts y_i to obtain $c_{y_i} = Enc_{k'_i}(y_i; M')$. It must also encrypt x_i as explained in the previous section. M' needs to be chosen large enough to prevent an overflow so it is set to $M' = n * t^2$. Each ciphertext c_{y_i} is therefore $log(M') = 2 * log(t) + log(n)$ bits long. The sensor forwards c_{y_i}, together with c_{x_i}, to its parent. The size of the resulting data is $3 * log(t) + 2 * log(n)$. The parent aggregates all the c_{y_j} of its k children by simply adding them up. It also aggregates,

separately, the c_{x_j}, as explained in the previous section. The two resulting values are then forwarded. The sink ends up with values C_x and $C_y = \sum_{i=1}^{n} c_{y_i} \bmod M$. C_x is used to compute the average Av. C_y is used to compute the variance as follows: The sink computes $V_x = Dec_{K'}(C_y; M') = C_y - K' \bmod M'$, where $K' = \sum_{i=1}^{n} k'_i$. The variance is then equal to $V_x/n - Av^2$.

5.2 Robustness

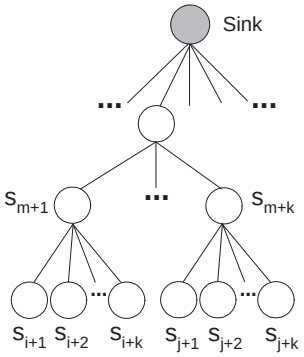


Fig. 1. Multi-level WSN model with nodes of degree k

An important consequence of using our proposed encryption scheme for aggregation in WSNs is that the sink node needs to be aware of the encryptors id's such that it can regenerate the correct keystream for decryption purposes.

Because WSNs are not always reliable, it cannot be expected that all nodes reply to all requests. Therefore there needs to be a mechanism for communicating the id's of the non-responding nodes to the base station. The simplest approach, and the one we used in our evaluation, is for the sensors to append their respective node id's to their messages[5].

6. SECURITY ANALYSIS

We use the CDA security model in [Chan and Castelluccia 2007] to analyze the concrete construction in Section 4. For completeness, the security model is given in Appendix A. As usual, the adversary is assumed to be probabilistic polynomial time (PPT) in the security model. In the model, the adversary can choose to compromise a subset of nodes and obtain all the secret information of these nodes. With oracle access, he can also obtain from any of the uncompromised nodes the ciphertext of any plaintext he chooses. The security goal is that the adversary cannot extract in polynomial time any information about the plaintext from a given ciphertext. This is the well known notion of semantic security [Goldwasser and Micali 1984]. Formally defined, the security model is described as a game in Appendix A.

[5]Depending on the number of nodes that respond to a query, it could be more efficient to communicate the id's of nodes that successfully reported values

The concrete construction in Section 4.3 can be shown to achieve semantic security or indistinguishability against chosen plaintext attacks (IND-CPA), an equivalent notion of semantic security [Goldwasser and Micali 1984], if the underlying key generation function is from a pseudorandom function family. The security can be summarized by the following theorem.

THEOREM 1. *The concrete construction is semantically secure against any collusion with at most* $(n-1)$ *compromised nodes (where* n *is the total number of nodes), assuming* $F_\lambda = \{f_s : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda\}_{s\in\{0,1\}^\lambda}$ *is a pseudorandom function and* $h : \{0,1\}^\lambda \rightarrow \{0,1\}^l$ *satisfies the requirement that* $\{t \leftarrow \{0,1\}^\lambda : h(t)\}$ *has a uniform distribution over* $\{0,1\}^l$.

Proof Sketch: Detailed proof is given in Appendix B. The basic idea is that we assume there exists a PPT adversary which can break the semantic security of the proposed encryption scheme, and then we show how this adversary can be used to break the indistinguishability property of the underlying pseudorandom function. By a contrapositive argument, we can say that if the pseudorandom function has the described indistinguishability property briefly described in Section 4.3, then the proposed encryption is semantically secure.[6] □

Note the standard security goal for encryption is indistinguishability against chosen ciphertext attacks in which the adversary is allowed to obtain the plaintext of any ciphertext of his choice [Naor and Yung 1990; Katz and Yung 2006]. If a stateful decryption mechanism is assumed, that is, the decryption algorithm keeps track of all the nonces previously used, our scheme could also be proved to be secure against chosen ciphertext attacks. But the resulting scheme might be inefficient. Nevertheless, it could still be practical since in usual situation only the sink would perform decryption. Since the aggregation functionality allows ciphertexts to be modified in some way without invalidating them, achieving chosen ciphertext security (more specifically, indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2)) with a stateless decryption mechanism may be impossible in this scenario.

7. OVERHEAD ANALYSIS

In this section, we compare the bandwidth of our proposed *AGG* protocol with the *No-Agg* (forwarding data packets) and HBH (hop-by-hop encryption and aggregation) approaches, as described in section 5. The overall bandwidth in the WSN and the number of bits sent by individual nodes are measured for different WSN tree like topologies. Below we describe the specific network model that we use in our measurements. The comparisons will be made for the two following cases: (1) the sink is only interested in the average value and (2) the sink is interested in the average and variance values.

7.1 Network Model

We envision a multi-level network tree in which there exist numerous sensor nodes and only one sink node. To simplify the model, we assume a balanced k-ary tree, as depicted in figure 1. Let t denote the range of possible measurement values collected by a sensor (i.e., if a sensor can measure temperatures between 0 and 99 Fahrenheit, then t = 100).

We will analyze the communication bandwidth in the proposed WSN model from two perspectives: (1) the number of bits sent per node at different levels in a 3-ary tree and (2) the total number of bits transmitted throughout the WSN for 3-ary trees of various height. These

[6] Formal details of the indistinguishability property of the pseudorandom function can be found in Appendix B.

measurements will be carried out for the three models that we are considering, namely *No-Agg*, HBH and AGG.

Next we describe how to calculate the number of bits (header and payload) sent per node for each of these schemes. We choose the packet format used in TinyOS [Karlof et al. 2004] which is the operating system running on the Berkeley motes that we envision as the sensor platform. The packet header is 56 bits and the maximum supported data payload is 232 bits.

For the *No-Agg* scheme, a node only needs $log(t)$ bits to encode its sensed data. In addition, all internal nodes need to forward the packets sent to them by their children, and the number of packets received grows exponentially (in k) as we move higher in the tree (i.e. closer to the sink).

In the HBH approach, the number of bits sent depends upon the node's level in the WSN tree. Leaf nodes only send $log(t)$ bits (same as in No-Agg), while nodes higher up in the tree will have aggregated data and therefore need to send more bits. Additionally, when the variance is also requested, the aggregating nodes need to keep track of this value separately, and use approximately $log(n't)$ bits to encode the value, where n' is the number of nodes aggregated so far.

With our AGG scheme, the number of bits sent by a node depends on the size of the modulus M used in the additive encryption scheme. Its size can be computed as the maximum possible aggregate value, which in this model turns out to be $M = n*t$, i.e. all sensors measure the largest possible reading. Therefore, when encoding the average, each node uses $log(M) = log(t)+log(n)$ bits. When the variance is also desired, a node needs to send the ciphertext corresponding to x^2. This requires an extra $log(n * t^2) = 2 * log(t) + log(n)$ bits. Additionally, an aggregator needs to append to the aggregate the id's of its children that did not reply to the query. These id's have to be propagated up to the sink along with the aggregate.

7.2 Numerical Results

In this section, we compare the performance of the *No-Agg*, *HBH* and AGG according to the following two criteria: (1) The forwarding cost per node i.e. the number of bits forwarded by node at each level of the delivery tree. (2) The overall bandwidth gain achieved by *HBH* and AGG over the *No-Agg* scheme.

Levels	Num Nodes	A (0%)	A (10%)	A (30%)	AV (0%)	AV (10%)	AV (30%)	HBH-A	HBH-AV	No-Agg
1	3	75	950	2700	100	975	2725	73	97	68859
2	9	75	366	950	100	392	975	72	94	22932
3	27	75	172	366	100	197	392	70	91	7623
4	81	75	107	172	100	132	197	68	87	2520
5	243	75	85	108	100	111	132	67	84	819
6	729	75	78	85	100	103	110	65	81	252
7	2187	75	75	75	100	100	100	63	63	63

Table I. Number of bits sent per node for each level in a 3-tree of depth 7, where the measured value range of 2^7.

Forwarding Cost per node (fairness). Table I shows the number of bits sent per node at each level in a 3-degree tree of height[7] 7 when $t = 128$ (the network is for example monitoring temperatures that range between 0 and 127 degrees).

For the *No-Agg* approach it becomes obvious from the results that there is a widely differing data communication load amongst sensors at different levels (nodes at level 7 send 3 orders of magnitude less data than those at level 1). Because the nodes closer to the sink have to send such significantly larger amounts of data than their descendants, they use up their batteries and die sooner. Should a level of nodes in the tree stop functioning, then the whole WSN stops functioning as well. Therefore, nodes would have to either be swapped around manually or replaced upon failure, both tasks being quite impractical when considering the number of nodes at the various levels.

The table shows a steady increase of bits per node for the HBH approach, both for the average (HBH-A) as well as the average and variance data (HBH-AV). Notice the relatively dramatic increase in bits transmitted between nodes at level 7 and 6 for HBH-AV. This is due to that the leaf nodes need not send a ciphertext representing x^2 (needed for the computation of the variance), where x represent their measured value, as it can be computed by their parents. Because packets are not forwarded as in No-Agg, we observe a significant reduction in bits sent per node at all non-leaf levels.

For the *AGG* we considered three scenarios: (1) all the nodes reply[8], (2) 90% of the nodes reply[9] and (3) 70% of the nodes reply[10]).

In the first scenario, there is a constant number of bits sent by each node at each level in the tree. However, this number of bits is larger than even the maximum for any HBH approach, due to the size of the modulus M. As previously explained, the number of bits sent by the leaves is larger with the aggregation methods (AGG-A: $56+log(t)+log(n) = 75$ bits, AGG-AV: $56 + 3 * log(t) + 2 * log(n) = 100$ bits) than when no aggregation is used ($56 + log(t) = 63$ bits). However, aggregation distributes the load evenly over all nodes, regardless of their distance to the sink. We believe this to be an attractive property in WSNs. In the second and the third scenarios, the number of bits processed by each node gets larger the closer it gets is to the sink. This is the result of appending the id's of the non-responding children to the aggregate. As we move up the tree the list of non-responding nodes increases. If we assume that x% of the total nodes do not participate, an aggregator A must append to the aggregated message, the id of x% of all its children (i.e of all the nodes in the subtree rooted at A). If, for example, A is at level 3 and $x = 30$, there are $3^4 = 81$ children and A has to append $81 * 0.3$ i.e. 25 ids. The total size of the aggregated message is then $63 + 25 * 12 = 366$ bits as shown in the table I.

Bandwidth Gain. Table II displays the bandwidth transmission gain of the HBH and AGG schemes over the *No-Agg* scheme using a 3-degree WSNs of various heights. We consider the gains when (1) only the average is computed and (2) both the average and variance are

[7]The sink is at level 0 in the tree

[8]Referred to in the tables as $A(0\%)$ when only the average is computed and as $AV(0\%)$ when the average and variance are computed.

[9]Referred in the table as $A(10\%)$ when only the average is computed and as $AV(10\%)$ when the average and variance are computed.

[10]Referred in the table as $A(30\%)$ when only the average is computed and as $AV(30\%)$ when the average and variance are computed.

Levels	Num Nodes	A (0%)	A (10%)	A (30%)	HBH-A	AV(0)	AV(10)	AV(30)	HBH-AV
3	40	2.42	2.39	2.34	2.58	1.89	1.87	1.84	2.24
4	121	3.20	3.13	3.01	3.50	2.46	2.40	2.37	3.02
5	364	3.96	3.82	3.6	4.46	3.03	2.98	2.84	3.84
7	3280	5.46	5.13	4.58	6.41	4.1	3.9	3.6	5.52
8	9841	6.22	5.72	4.95	7.39	4.59	4.3	3.85	6.37

Table II. WSN bandwidth performance gain of the AGG and HBH schemes when aggregating the (1) Average and (2) Average and Variance for a 3-tree and $t = 2^7 = 128$.

computed [11]. These gains are obtained by computing the total bandwidth costs, C_{HBH}, C_{AGG} and C_{No-Agg}, by adding, for each of these schemes, the total number of bits forwarded by each node of the network. The bandwidth gain of HBH and AGG are respectively defined as C_{No-Agg}/C_{HBH} and C_{No-Agg}/C_{AGG}.

For example, in a 3-tree of height 5, there are 364 nodes, and when only computing the average value, *AGG-A* achieves a factor of 3.96 speedup over No-Agg, i.e. approximately 4 times less bits are sent across the network. As expected, *HBH-A* and *HBH-AV* have better performance than both *AGG-A* and *AGG-AV*, respectively, although they both outperform No-Agg. The biggest draw for using *AGG* over HBH is that of end-to-end privacy. With HBH, it is enough for an attacker to compromise one node close to the sink to gain a large picture of the aggregated data in the WSN. This is because each node in HBH stores the secret key needed for decryption (and encryption), leaving them vulnerable. On the other hand, nodes in *AGG* do not store sensitive key material and the only data an attacker can learn is a single sensor's individual reading.

The results shown in this section are very encouraging since they confirm that aggregation is a useful technique for reducing the total bandwidth usage and can therefore extend the overall lifetime of the network.

7.3 Computational Costs

This section discusses the computation costs of the proposed scheme and the issues related to implementation. Let t_{add} and t_{multi} respectively denote the cost of an addition and a multiplication operation in mod M. Note that the cost of performing a subtraction is also t_{add}. Let t_{prf} and t_h denote the costs of evaluating an instance of a certain pseudorandom function and a length-matching hash function respectively. Let t_{ce} and t_{cd} denote the costs of running one encryption and one decryption of a particular cipher used in the hop-by-hop approach. The overall computation costs for the proposed protocols are depicted below, assuming L reporting nodes, i.e. $|hdr| = L$. For the aggregation operation, the calculations assume each aggregation involves only two inputs. The computation costs are summarized by the following table.

The aggregate encryption scheme places all the decryption computation tasks at the sink while the hop-by-hop scheme distributes the decryption cost over all the aggregating nodes in the network. Hence, a sensor device may need to perform more computation than the sink in the hop-by-hop approach. Since the sink is usually a more powerful device, the aggregate encryption approach could be preferable in the WSN scenario.

In the aggregate encryption scheme, each node only needs to perform one evaluation of

[11] We remind the reader of that in the *No-Agg* scheme, no extra values need to be sent when the variance needs to be computed.

	Hop-by-hop Encryption (HBH)	Aggregate Encryption (AGG)
Encryption	t_{ce}	$t_{prf} + t_h + t_{add}$
Decryption	t_{cd}	$2L \cdot t_{prf} + L \cdot t_h + L \cdot t_{add}$
Aggregation (per 2 inputs)	$2 \cdot t_{cd} + t_{ce} + t_{add}$	t_{add}

Table III. A comparison of computation costs.

the underlying pseudorandom function and one evaluation of the length-matching hash and one addition mod M for encryption and one addition for aggregation. Aggregation is pretty efficient in the aggregate encryption scheme. If the hash is implemented by bit truncation, the computation cost is negligible compared to that of an addition operation. If the hash is implemented by truncation combined with exclusive-OR, the computation cost is of the same order as the cost of an addition operation. We could thus say the cost of evaluating h can be neglected in the calculation of the overall computation cost of encryption without much loss of accuracy. As a result, the cost for encryption is dominated by the cost of evaluating one instance of the pseudorandom function.

As mentioned in Section 4.3, a collision resistant hash can be used in place of a number-theory based pseudorandom function for key derivation if its assumed pseudorandomness is acceptable for an application. [Perrig et al. 2001] shows an example of such usage in sensor networks and demonstrates its feasibility in terms of computation complexity. Hence, the computation cost of the proposed aggregate encryption scheme should be reasonable for most WSN applications.

8. AGGREGATE AUTHENTICATION AGAINST OUTSIDER-ONLY ATTACKS

While the aggregate encryption given in this paper could provide good end-to-end privacy, like any data aggregation over plaintexts such as [Madden et al. 2002], it is vulnerable to false data injection attacks. In its current form, even an external attacker can add an arbitrary value as his contribution to an aggregate ciphertext in the encryption scheme in Section 4.3.

The aggregate encryption scheme is complementary to most authentication techniques in the literature including [Chan et al. 2006; Yang et al. 2006; Przydatek et al. 2003; Hu and Evans 2003]; hence, these techniques can be added as plug-ins to the proposed aggregate encryption. It should be noted that all these techniques are not end-to-end, namely, some kind of call-backs to the aggregating nodes (after the sink receives the aggregate) are necessary. This section provides an end-to-end alternative to aggregate authentication but it can only guard against an external attacker without knowledge of a secret group key; the existence of any compromised nodes would imply a total breath of security.

A general application scenario in which such an aggregate authentication scheme would be useful is as follows: Suppose during the deployment of a sensor network, physical captures of nodes are not possible, say, the network is deployed in an inaccessible area (such as an accident scene inside a nuclear reactor or a bush fire scene) or not unattended (for instance, guarded by a team for under-sea terrain surveying). But part of the communication over the network (say close to the sink or collecting point) can be eavesdropped or manipulated without going inside the scene during its deployment. The proposed aggregate authentication scheme can provide sufficient integrity protection during deployment since no compromised node can be assumed. However, after the task is complete, the sensor devices may be disposed of on the spot due to the difficulty of recalling. An adversary might then be able to obtain some of the sensor devices (say, after a long enough period of time since a nuclear accident). Combined with previously

recorded messages eavesdropped from the network, the adversary might be able to recover part or even all of the previously protected communication. Hence, a stronger privacy protection scheme is needed. Under-sea terrain surveying and resource exploration could be examples of this scenario.[12] In summary, a scheme providing strong privacy with integrity protection against outsider-only attacks could find applications in scenarios where physical captures of sensor devices are not possible during deployment but could be achievable after deployment.

In [Chan and Castelluccia 2008], the notion of aggregate message authentication codes (AMAC) is proposed as a natural extension of one-to-one message authentication codes (MAC) and it is shown that no scheme can be designed to achieve such a notion. Since the proposed notion is not a contrived one, it could be fair to say no scheme may be constructed to provide end-to-end aggregate integrity against chosen message attacks in the presence of compromised nodes.

Even with call-backs, say in [Chan et al. 2006], the only guarantee provided so far is that an adversary cannot manipulate an aggregation result by an amount beyond what can be achieved through data injection at the compromised nodes unless prior knowledge of the statistical distribution of the data is utilized for outlier detection at the sink. In the context of additive aggregation without requesting each sensor to provide a range proof on its contribution, the impact of a compromised node in [Chan et al. 2006] (regarding manipulation of an aggregation result) is essentially the same as its counterpart in our proposed aggregate authentication scheme. Indeed, range proving requires prior knowledge of the statistical distribution of data.

When there is no compromised node, our scheme assures that no data can be injected into an aggregate without being detected. The basic idea of our scheme is to add to each node's data a keyed, aggregatable checksum the computation of which requires the knowledge of a shared group key. Without knowledge of the group key, it would be a difficult task for an external attacker to compute a valid checksum for any modified data. It should be noted that compromising any one of the nodes (and hence the group key) would cause a complete security breach of the authentication scheme. Nevertheless, this could be the best one can achieve for end-to-end aggregate authentication.

8.1 Details of the Protocol

The details of the aggregate authentication protocol are as follows.

[12]Suppose there are two conflicting, neighboring countries, A and B, with overlapping Exclusive Economic Zone (EEZ). They both claim possession of a certain under-sea oil field. Country A, which is more technologically advanced, sends out an exploration sensor network to measure the under-sea terrain and related data to prepare for oil exploitation in the near future. Country B is supposed not to be able to make such exploration but wants to eavesdrop the data collected by country A for its own exploitation. In order to protect its own interests, country A would wish to ensure long-term privacy of the measured exploration data since an exploitation project may last for a decade or so. Since an exploration team is usually close to the deployed sensor network in usual practice and the sensor devices (deep under sea) are inaccessible, so no physical capture of sensor devices can be assumed. Authentication against outsider-only attacks is thus sufficient when the network is not unattended. However, once the exploration job is completed, the sensor devices will be left unattended due to the high cost of recalling or post-deployment guarding. As the network stops operating, some devices may be drifted by sea waves and current and floats up to the surface. These devices may then be captured by country B. Privacy against compromised nodes is therefore deemed as needed.

Combined Encryption and Aggregate Authentication Scheme

Key Distribution:
Each sensor node i is given 3 secret keys (k_i, k_i', k). They can be generated from three independent master keys using a pseudorandom function as in the basic scheme. The sink keeps all the three master keys. k_i and k_i' correspond to the encryption key ek_i in the basic scheme. Each node should receive a distinct pair of (k_i, k_i') while getting a common group key k.

Encryption + Checksum Computation:
Let M be the modulus of the arithmetics. For an arbitrary reporting epoch r.

(1) Each node i generates the session keys $(k_i^{(r)}, k_i^{(r)'}, k^{(r)})$ from its secret keys (k_i, k_i', k) using a pseudorandom function and the length-matching hash function as in the basic scheme. (i.e. $k_i^{(r)} = h(f_{k_i}(N_r))$, $k_i^{(r)'} = h(f_{k_i'}(N_r))$, and $k^{(r)} = h(f_k(N_r))$ where $f(\cdot)$ is the pseudorandom function used, $h(\cdot)$ is the length-matching hash function and N_r is the nonce used for epoch r.)
(2) For a plaintext message $m_i \in [0, M-1]$, encrypt m_i using $k_i^{(r)}$ to obtain the ciphertext $x_i = m_i + k_i^{(r)} \bmod M$.
(3) Compute the checksum: $y_i = m_i \cdot k^{(r)} + k_i^{(r)'} \bmod M$.
(4) The ciphertext and checksum is: (hdr, x_i, y_i) where $hdr = \{i\}$.

Decryption + Verification:

(1) Given a ciphertext (hdr, x, y), generate the session keys $(k_i^{(r)}, k_i^{(r)'}, k^{(r)})$ for each $i \in hdr$.
(2) Compute $m = x - \sum_{i \in hdr} k_i^{(r)} \bmod M$. m is the decrypted plaintext.
(3) Check $y \stackrel{?}{=} \sum_{i \in hdr} k_i^{(r)'} + k^{(r)} \cdot m \bmod M$. If yes, set $b = 1$, otherwise, set $b = 0$.
(4) Return (m, b). Note that $b = 0$ indicates a verification failure.

Addition of Ciphertexts: Given two ciphertexts (hdr_i, x_i, y_i) and (hdr_j, x_j, y_j),

(1) Compute $hdr_l = hdr_i \cup hdr_j$.
(2) Compute $x_l = x_i + x_j \bmod M$.
(3) Compute $y_l = y_i + y_j \bmod M$.
(4) The aggregated ciphertext is: (hdr_l, x_l, y_l).

The final aggregated ciphertext (hdr, x, y) received at the sink can be expressed as two equations:

$$\begin{cases} x &= K_1^{(r)} + m \\ y &= K_2^{(r)} + K^{(r)} \cdot m \end{cases} \quad (1)$$

where m is the final aggregate of the plaintext data and $K_1^{(r)}, K_2^{(r)}, K^{(r)}$ are two sums of node keys and the common group key (for epoch r) given by the following expressions:

$$K_1^{(r)} = \sum_{i \in hdr} k_i^{(r)}, \; K_2^{(r)} = \sum_{i \in hdr} k_i^{(r)'}, \text{ and } K^{(r)} = k^{(r)}.$$

Equation (1) can be viewed as a set of constraint equations (for a particular hdr) that a

correct pair (x, y) should satisfy. For each epoch, *hdr* is part of the input to the verification process to define the coefficients $K_1^{(r)}, K_2^{(r)}, K^{(r)}$ of the constraint equations in (1); *hdr* uniquely specifies a subset of nodes whose data are supposed to have been incorporated in (x, y).

If (x, y) has not been tampered with, the plaintext aggregate m extracted from the first constraint equation in (1) should satisfy the second constraint equation in (1); m is a correct aggregate of the data contributed by the nodes in *hdr* when they all act honestly. The goal of an external adversary is thus to find a different valid pair (x', y') for the same *hdr* such that

$$\begin{cases} x' &= K_1^{(r)} + m' \\ y' &= K_2^{(r)} + K^{(r)} \cdot m' \end{cases}$$

for some $m' \neq m$ and m' is not necessarily known by the adversary. Note that the coefficients $K_1^{(r)}, K_2^{(r)}, K^{(r)}$ have to be the same as that in the equations for (x, y) for a successful forgery. Without knowledge of $K^{(r)}$, the probability for any PPT adversary to find such a valid pair (x', y') for the given *hdr* should be negligibly small. The proposed protocol guarantees with high probability that, for an epoch r, any pair (x, y) which passes the verification test for a given *hdr* has to allow the recovery of a correct aggregate whose contributions can only come from nodes in *hdr* with knowledge of $K^{(r)}$ (with exactly one contribution from each node in *hdr*).

In any epoch, by passively observing transmissions from honest nodes in a network, an adversary without knowledge of $K^{(r)}$ can still create multiple tuples of the form (hdr, x, y), each with a *distinct hdr*, to pass the verification test of Equation (1). This can be achieved by simply aggregating valid ciphertext-checksum pairs eavesdropped in the transmissions of the honest nodes. However, it should be noted that, for each *hdr*, there is at most one such tuple and the corresponding pair of (x, y) is indeed a correct ciphertext-checksum pair for *hdr* in the sense that this pair of (x, y), upon verification, can recover an aggregate m the contributions of which only originate from the honest nodes specified in *hdr*, that is, $m = \sum_{i \in hdr} m_i$ where m_i is the measurement of node i. In other words, in the set $\mathcal{C}$ of ciphertext-checksum pairs obtained by combining eavesdropped pairs through the aggregation functionality, if a pair $(x, y) \in \mathcal{C}$ passes the verification equations in (1) for *hdr*, any pair $(x', y') \in \mathcal{C}$ which can satisfy the same set of equations (i.e. with the same set of coefficients) has to be equal to (x, y). Hence, any external attacker without knowledge of $K^{(r)}$ still cannot inject its data into an aggregate ciphertext pair (x, y) which satisfies the constraint equations in (1) even though he may be able to create such a pair from the ciphertext-checksum pairs obtained from eavesdropping the transmissions of honest nodes; neither can the attacker modify an existing valid pair of (x, y) to pass the verification test for the same *hdr* but produce a different aggregate output except with a negligibly small probability.[13]

It is thus fair to say the best that an external adversary without knowledge of $K^{(r)}$ can achieve in breaking the proposed scheme is essentially limited to excluding the contributions of some honest nodes from being incorporated into an aggregate. Such exclusion would usually have slight impact in the calculation of mean and variance unless the exclusion makes up a pretty large fraction of nodes in which case it would make the sink suspect the occurrence of a possible attack. It should be emphasized that, to achieve so with impact, the adversary must be capable to intercept and modify a considerable portion of the transmissions in the entire network, which is normally hard for an attacker to achieve.

[13] An adversary may be able to obtain another valid (x, y) pair but it is valid only for a different *hdr*.

8.2 Security Analysis

Recall that the goal of the proposed extension of aggregate authentication is to guard against any external attackers (without knowledge of the keys) from injecting data into an aggregate. The security of the proposed scheme is summarized by the following theorem.

THEOREM 2. *Given a genuine ciphertext-checksum pair (x, y) corresponding to an aggregate m which incorporates data from a group of nodes specified by hdr and all other communication transcripts between nodes, the probability of successfully forging a valid pair $(x', y') \neq (x, y)$ for some $m' \neq m$ to pass the verification test of the aggregate authentication scheme for the same hdr is negligible for any external PPT (Probabilistic Poly-Time) adversary without knowing K, assuming the encryption keys and the group key are generated by a pseudorandom function based on different seed keys.*

Proof: Assume the pseudorandom function has some indistinguishability property as usual. We prove by contradiction, showing that a PPT adversary which can forge a valid pair (x', y') can also break the indistinguishability property of the underlying pseudorandom function. We show the reduction[14] in two steps: first, we show that a forging algorithm to find (x', y') can be used as a sub-routine to solve a newly defined problem called "Under-determined Equation Set with Pseudorandom Unknowns (UESPU)"; then we show that the UESPU problem is computationally hard if the underlying pseudorandom function has the usual indistinguishability property. The UESPU problem is defined as follows:

Under-determined Equation Set with Pseudorandom Unknowns (UESPU) Problem — Suppose K_1, K_2, K are independent random seeds. Let $K_1^{(r)}$, $K_2^{(r)}$ and $K^{(r)}$ denote the hashed outputs of a pseudorandom function f at input r corresponding to seed keys K_1, K_2 and K.[15] Given a 3-tuple (m, x, y) where $x = K_1^{(r)} + m$ and $y = K_2^{(r)} + K^{(r)} \cdot m$, find $(K_1^{(r)}, K_2^{(r)}, K^{(r)})$ while allowed to evaluate the pseudorandom function at any input $r' \neq r$.[16]

Without loss of generality, in the UESPU problem, each of $K_1^{(r)}$, $K_2^{(r)}$ and $K^{(r)}$ is treated as a single hashed output of f. In the proposed aggregate authentication, they are the sums of hashed outputs of f. If they are represented as the sums of hashed output of f instead, the modified problem would remain hard if f is a pseudorandom function.

Solving the UESPU problem using a forger of (x', y').

Suppose there exists a PPT adversary $\mathcal{A}$ which can forge a valid pair (x', y') at an epoch with nonce r with non-negligible probability p_f. Using $\mathcal{A}$ as a subroutine, we can construct another algorithm $\mathcal{A}'$ to find $(K_1^{(r)}, K_2^{(r)}, K^{(r)})$ from (m, x, y) with probability p_f in any instance of the

[14]The reduction of the problem of breaking the indistinguishability of the pseudorandom function to the problem of forging a valid (x', y') pair.

[15]That is, $K_1^{(r)} = h(f_{K_1}(r))$, $K_2^{(r)} = h(f_{K_2}(r))$, and $K^{(r)} = h(f_K(r))$ where h is the length-matching hash function.

[16]The UESPU problem is typically hard if f is a pseudorandom function. More formally defined, given that l is the key length of the pseudorandom function f and h is a length-matching hash function, the following probability is negligible in l for any PPT algorithm $\mathcal{A}$.

$$Pr\left[\begin{array}{l} K_1 \leftarrow \{0,1\}^l; K_2 \leftarrow \{0,1\}^l; K \leftarrow \{0,1\}^l; r \leftarrow \{0,1\}^l; \\ K_1^{(r)} = h(f_{K_1}(r)); K_2^{(r)} = h(f_{K_2}(r)); K^{(r)} = h(f_K(r)); \ : \ \mathcal{A}^f(m,x,y) = (K_1^{(r)}, K_2^{(r)}, K^{(r)}) \\ m \leftarrow \mathbb{Z}_M; x = K_1^{(r)} + m; y = K_2^{(r)} + K^{(r)} \cdot m \end{array}\right]$$

UESPU problem. Note that $\mathcal{A}'$ should be able to answer queries from $\mathcal{A}$ for any $r' \neq r$ by passing the queries to its challenger.

The construction of $\mathcal{A}'$ is as follows: Give $\mathcal{A}$ the pair (x, y). When $\mathcal{A}$ returns a pair $(x', y') \neq (x, y)$, we can determine $K_1^{(r)}$, $K_2^{(r)}$, $K^{(r)}$ from the resulting set of equations. The explanation is as follows:
Note that

$$\begin{aligned} x &= K_1^{(r)} + m \\ y &= K_2^{(r)} + K^{(r)} \cdot m. \end{aligned}$$

So we have two equations and 3 unknowns. If (x', y') is a valid forgery, then it must satisfy the following two equations (with the same $K_1^{(r)}$, $K_2^{(r)}$ and $K^{(r)}$) in order to pass the verification test:

$$\begin{aligned} x' &= K_1^{(r)} + m' \\ y' &= K_2^{(r)} + K^{(r)} \cdot m' \end{aligned}$$

for some unknown value $m' \neq m$.

The pair (x', y') adds in two new equations and one unknown m'. Since $(x', y') \neq (x, y)$ and $m' \neq m$, it can be assured that the four equations are independent. Hence, there are four independent equations and four unknowns in total and it should be easy to solve for $K_1^{(r)}$, $K_2^{(r)}$, $K^{(r)}$ (a contradiction to the UESPU assumption). The probability of solving the problem in the UESPU assumption is hence p_f.

Suppose there are n reporting nodes. The communication transcripts can be simulated easily by randomly picking $(n-1)$ random pairs of ciphertext-checksum (x_i, y_i) and subtracting them from (x, y) to obtain the n-th pair. Since $\mathcal{A}$ does not have any knowledge about the node keys, real pairs of (x_i, y_i) should look random to $\mathcal{A}$. Hence, $\mathcal{A}$ could not distinguish its view in the simulation and that in the real attack. On the other hand, it could be concluded that knowing (x_i, y_i) without knowing the node keys would not help in creating a valid forgery. In the above discussion, we treat $K_1^{(r)}$, $K_2^{(r)}$, $K^{(r)}$ as a single output of a pseudorandom function for the sake of clarity and easy comprehension; more precisely, in the aggregate authentication scheme, each one of them is the sum of outputs of a pseudorandom function seeded with distinct keys (one from each sensor node). Nonetheless, the above arguments and conclusion apply to both cases.

A distinguisher for the pseudorandom function using an algorithm which solves the UESPU problem.

The UESPU problem is hard if $K_1^{(r)}$, $K_2^{(r)}$, $K^{(r)}$ are generated by a pseudorandom function. Obviously, m and x can uniquely determine $K_1^{(r)}$. But the equation $y = K_2^{(r)} + K^{(r)} \cdot m$ has two unknowns, which cannot be uniquely determined. It could be shown that if there exists an algorithm $\mathcal{A}'$ solving in poly-time $K_2^{(r)}$ and $K^{(r)}$ from m and y, then the indistinguishability property of the underlying pseudorandom function is broken.

The idea is as follows: assume the seed key for generating $K^{(r)}$ is unknown but the seed key for generating $K_2^{(r)}$ is known. That is, we can generate $K_2^{(r')}$ for any r'. When a challenge $K^{(r)}$ is received, we have to determine whether it is randomly picked from a uniform distribution or generated by the pseudorandom function with an unknown seed key. We generate $K_2^{(r)}$ from the known seed key. Then we pass $y = K_2^{(r)} + K^{(r)} \cdot m$ to $\mathcal{A}'$. If the solution from $\mathcal{A}'$ does not

match the generated $K_2^{(r)}$, we reply that $K^{(r)}$ is randomly picked, otherwise, it is generated from the pseudorandom function. If $\mathcal{A}'$ has non-negligible probability of breaking the UESPU assumption, the above construction would also has a non-negligible advantage of breaking the indistinguishability property of the underlying pseudorandom function. Note that all queries from $\mathcal{A}'$ could be answered by sending queries to the challenger and running the pseudorandom function with the known key.

□

8.3 Additional Overheads

The aggregate authentication extension leads to additional costs in both communication and computation. For the communication cost, the length of each ciphertext is now increased by $|M|$ (where M is the modulus of the arithmetics in use). This is the size of the added checksum. For the computation cost, the notations of Section 7.3 are used. The additional computation costs needed for checksum generation and verification are summarized as follows. In the calculation of verification cost, the cost of a comparison operation in mod M is assumed similar to the cost of an addition operation in mod M.

	Additional Computation Costs
Checksum Generation	$2 \cdot t_{prf} + 2 \cdot t_h + t_{add} + t_{multi}$
Checksum Verification	$(2L+1) \cdot t_{prf} + (L+1) \cdot t_h + (L+1) \cdot t_{add} + t_{multi}$

Table IV. Additional computation costs of the extension of aggregate authentication (assuming L is the number of nodes contributing to an aggregate).

9. RELATED WORK

The problem of aggregating encrypted data in WSNs was partially explored in [Girao et al. 2004]. In this paper, the authors propose to use an additive and multiplicative homomorphic encryption scheme to allows aggregation of encrypted data. While this work is very interesting, it has several important limitations. Firstly, it is not clear how secure the encryption scheme really is. Secondly, as acknowledged by the authors, the encryption and aggregation operations are very expensive and therefore require quite powerful sensors. Finally, in the proposed scheme, the encryption expands the packet size significantly. Given all these drawbacks, it is questionable whether aggregation is still beneficial. In contrast, our encryption scheme is proven to be secure and is very efficient. Encryption and aggregation only requires a small number of single-precision additions. Furthermore, our encryption scheme only expands packet sizes by a small number of bits. As a result, it is well adapted to WSNs consisting of very resource constrained sensors.

In [Hu and Evans 2003], Hu and Evans propose a protocol to securely aggregate data. The paper presents a way to aggregate MACs (message authentication code) of individuals packets such that the sink can eventually detects non-authorized inputs. This problem is actually complementary to the problem of aggregating encrypted data, we are considering in this paper. The proposed solution introduces significant bandwidth overhead per packet. Furthermore, it requires the sink to broadcast n keys, where n is the number of nodes in the network, at each sampling period. This makes the proposed scheme non-practical.

Although not related to data privacy, in [Przydatek et al. 2003] Przydatek, et al. present efficient mechanism for detecting forged aggregation values (min, max, median, average and count). In their setting, a trusted outside user can query the WSN. The authors then look into how to reduce the trust placed in the sink node (base station) while ensuring correctness of the query response. Another work by Wagner [Wagner 2004] examines security of aggregation in WSNs, describing attacks against existing aggregation schemes before providing a framework in which to evaluate such a scheme's security.

10. CONCLUSION

This paper proposes a new homomorphic encryption scheme that allows intermediate sensors (aggregators) to aggregate the encrypted data of their children without having to decrypt them. As a result, even if an aggregator gets compromised, the attacker won't be able to eavesdrop on the data and aggregate, resulting in much stronger privacy than an aggregation scheme relying on by hop-by-hop encryption.

We show that if the key streams used in our scheme are derived using a pseudorandom function, our scheme can achieve semantic security against any collusion of size less than the total number of nodes.

We evaluate the performance of our scheme. We show, as expected, that our scheme is slightly less bandwidth efficient than the hop-by-hop aggregation scheme described previously. However it provides a much stronger level of security. The privacy protection provided by our scheme is in fact comparable to the privacy protection provided by a scheme that would use end-to-end encryption and no aggregation (i.e. the aggregation is performed at the base station). We show that our scheme is not only much more bandwidth-efficient than such an approach, but it also distributes the communication load more evenly amongst the network nodes, resulting in an extended longevity of the WSN.

Finally, we extend our scheme to provide end-to-end aggregate authentication. Without knowledge of a group key, an external attacker has negligible probability of tampering the aggregate without being detected in the extension.

In conclusion, we give efficient, provably secure solutions to provide end-to-end privacy and authenticity (with reasonably good security assurance) for WSNs while en-route aggregation is supported. The presented scheme only supports mean and variance computation. However, we shown in [Castelluccia and Soriente 2008] that our construction could be used as a building block for other aggregation schemes to support more functions (such as medium, mode, range,...).

REFERENCES

Bellare, M., Canetti, R., and Krawczyk, H. 1996. Keying hash functions for message authentication. In *Advances in Cryptology — CRYPTO 1996, Springer-Verlag LNCS vol. 1109*. 1–15.

Boneh, D., Gentry, C., Lynn, B., and Shacham, H. 2003. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology — EUROCRYPT 2003, Springer-Verlag LNCS vol. 2656*. 416–432.

Castelluccia, C., Mykletun, E., and Tsudik, G. 2005. Efficient aggregation of encrypted data in wireless sensor networks. In *the Proceedings of MobiQuitous'05*. 1–9.

Castelluccia, C. and Soriente, C. 2008. ABBA: Secure aggregation in wsns - a bins and balls approach. *6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*.

Chan, A. C.-F. and Castelluccia, C. 2007. On the privacy of concealed data aggregation. In *ESORICS 2007, Springer-Verlag LNCS vol. 4734*. 390–405.

Chan, A. C.-F. and Castelluccia, C. 2008. On the (im)possibility of aggregate message authentication codes. ePrint Archive, Report 2008-. http://.

Chan, H., Perrig, A., and Song, D. 2006. Secure hierarchical in-network aggregation in sensor networks. In *ACM Conference on Computer and Communication Security (CCS 06)*. 278–287.

Eschenauer, L. and Gligor, V. D. 2000. A key management scheme for distributed sensor networks. *ACM CCS*, 41–47.

Girao, J., Westhoff, D., and Schneider, M. 2004. CDA: Concealed data aggregation in wireless sensor networks. *ACM WiSe 2004*.

Goldreich, O. 2001. *Foundations of Cryptography: Part 1*. Cambridge University Press.

Goldwasser, S. and Micali, S. 1984. Probabilistic encryption. *Journal of Computer and System Sciences 28,* 2, 270–299.

Goldwasser, S., Micali, S., and Rivest, R. 1988. A secure signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing 17,* 2, 281–308.

Hu, L. and Evans, D. 2003. Secure aggregation for wireless networks. *Workshop on Security and Assurance in Ad hoc Networks*.

Iwata, T. and Kurosawa, K. 2003. OMAC: One-key CBC MAC. In *Fast Software Encryption (FSE 2003), Springer-Verlag LNCS vol. 2887*. 129–153.

Karlof, C., Sastry, N., and Wagner, D. 2004. Tinysec: a link layer security architecture for wireless sensor networks. *Embedded Networked Sensor Systems*, 162–175.

Karlof, C. and Wagner, D. 2003. Secure routing in wireless sensor networks: Attacks and countermeasures. *Workshop on Sensor Network Protocols and Applications*.

Katz, J. and Yung, M. 2006. Characterization of security notions for probabilistic private-key encryption. *Journal of Cryptology 19,* 1, 67–95.

Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2002. TAG: a Tiny AGgregation service for ad-hoc sensor networks. *Fith Annual Symposium on Operating Systems Design and Implementation*, 131–146.

Naor, M., Reingold, O., and Rosen, A. 2002. Pseudorandom functions and factoring. *SIAM Journal on Computing 31,* 5, 1383–1404.

Naor, M. and Yung, M. 1990. Public-key cryptosystems provably secure against chosen-ciphertext attacks. In *ACM Symposium on Theory of Computing (STOC 1990)*. 427–437.

NIST. 2001. Advanced encryption standard. *NIST (National Institute of Standards and Technology) FIPS PUB 197*.

Perrig, A., Stankovic, J., and Wagner, D. 2004. Security in wireless sensor networks. *Communications of the ACM 47*, 53–57.

Perrig, A., Szewczyk, R., Wen, V., Culler, D., and Tygar, D. 2001. SPINS: Security protocols for sensor networks. In *the Proceedings of ACM MOBICOM 2001*. 189–199.

Przydatek, B., Song, D., and Perrig, A. 2003. SIA: Secure information aggregation in sensor networks. *ACM SENSYS*, 255–265.

Rivest, R. L. 1995. The RC5 encryption algorithm. *Dr. Dobb's Journal 1008*.

Rivest, R. L., Shamir, A., and Adleman, L. M. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM 21*, 120–126.

Vernam, G. S. 1926. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Journal of the American Institute of Electrical Engineers 45*, 105–115. See also US patent #1,310,719.

Wagner, D. 2004. Resilient aggregation in sensor networks. *Workshop on Security of Ad Hoc and Sensor Networks*.

Westhoff, D., Girao, J., and Acharya, M. 2006. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaption. *IEEE Transactions on Mobile Computing 5,* 10, 1417–1431.

Wood, A. D. and Stankovic, J. A. 2002. Denial of service in sensor networks. *IEEE Computer 35*, 54–62.

Yang, Y., Wang, X., Zhu, S., and Cao, G. 2006. SDAP: A secure hop-by-hop data aggregation protocol for sensor networks. In *the Proceedings of ACM Internation Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) 2006*.

Zhu, S., Setia, S., Jajodia, S., and Ning, P. 2004. An interleaved hop-by-hop authentication scheme for filtering false data in sensor networks. *IEEE Symposium on Security and Privacy*.

Appendix A: Semantic Security of Concealed Data Aggregation (CDA) [Chan and Castelluccia 2007]

Notation

We follow the notations for algorithms and probabilistic experiments that originate in [Goldwasser et al. 1988]. A detailed exposition can be found there. We denote by $z \leftarrow A(x, y, \ldots)$ the experiment of running probabilistic algorithm A on inputs $x, y \ldots$, generating output z. We denote by $\{A(x, y, \ldots)\}$ the probability distribution induced by the output of A. The notations $x \leftarrow \mathcal{D}$ and $x \in_R \mathcal{D}$ are equivalent and mean randomly picking a sample x from the probability distribution $\mathcal{D}$; if no probability function is specified for $\mathcal{D}$, we assume x is uniformly picked from the sample space. We denote by $\mathbb{N}$ the set of non-negative integers. As usual, PPT denote probabilistic polynomial time. An empty set is always denoted by ϕ.

CDA Syntax

A typical CDA scheme includes a sink R and a set U of n source nodes (which are usually sensor nodes) where $U = \{s_i : 1 \leq i \leq n\}$. Denote the set of source identities by ID; in the simplest case, $ID = [1, n]$. In the following discussion, $hdr \subseteq ID$ is a header indicating the source nodes contributing to an encrypted aggregate. A source node i has the encryption key ek_i while the sink keeps the decryption key dk from which all ek_i's can be computed. Given a security parameter λ, a CDA scheme consists of the following polynomial time algorithms.

Key Generation (KG). Let $\mathsf{KG}(1^\lambda, n) \rightarrow (dk, ek_1, ek_2, \ldots, ek_n)$ be a probabilistic algorithm. Then, ek_i (with $1 \leq i \leq n$) is the encryption key assigned to source node s_i and dk is the corresponding decryption key given to the sink R.

Encryption (E). $\mathsf{E}_{ek_i}(m_i) \rightarrow (hdr_i, c_i)$ is a probabilistic encryption algorithm taking a plaintext m_i and an encryption key ek_i as input to generate a ciphertext c_i and a header $hdr_i \subset ID$. Here hdr_i indicates the identity of the source node performing the encryption; if the identity is i, then $hdr_i = \{i\}$. *Sometimes the encryption function is denoted by* $\mathsf{E}_{ek_i}(m_i; r)$ *to explicitly show by a string* r *the random coins used in the encryption process.*

Decryption (D). Given an encrypted aggregate c and its header $hdr \subseteq ID$ (which indicates the source nodes included in the aggregation), $\mathsf{D}_{dk}(hdr, c) \rightarrow m/\perp$ is a deterministic algorithm which takes the decryption key dk, hdr and c as inputs and returns the plaintext aggregate m or possibly $\perp$ if c is an invalid ciphertext.

Aggregation (Agg). With a specified aggregation function f such as additive aggregation considered in this paper, $\mathsf{Agg}_f(hdr_i, hdr_j, c_i, c_j) \rightarrow (hdr_l, c_l)$ aggregates two encrypted aggregates c_i and c_j with headers hdr_i and hdr_j respectively (where $hdr_i \cap hdr_j = \phi$) to create a combined aggregate c_l and a new header $hdr_l = hdr_i \cup hdr_j$. Suppose c_i and c_j are the ciphertexts for plaintext aggregates m_i and m_j respectively. The output c_l is the ciphertext for the aggregate $f(m_i, m_j)$, namely, $\mathsf{D}_{dk}(hdr_l, c_l) \rightarrow f(m_i, m_j)$. This paper considers $f(m_i + m_j) = m_i + m_j \bmod M$. *Note that the aggregation algorithm does not need the decryption key* dk *or any of the encryption keys* ek_i *as input; it is a public algorithm.*

It is intentional to include the description of the header hdr in the security model to make it as general as possible (to cover schemes requiring headers in their operations). hdr is needed in some schemes to identify the set of decryption keys required to decrypt a certain ciphertext.

Nonetheless, generating headers or including headers as input to algorithms should not be treated as a requirement in the actual construction or implementation of CDA algorithms. For constructions which do not need headers, all hdr's can simply be treated as the empty set ϕ in the security model.

The Notion of Semantic Security

Only one type of oracle queries (adversary interaction with the system) is allowed in the security model, namely, the encryption oracle $\mathcal{O}_E$. The details are as follows:

Encryption Oracle $\mathcal{O}_E(i, m, r)$.. For fixed encryption and decryption keys, on input an encryption query $\langle i, m, r\rangle$, the encryption oracle retrieves s_i's encryption key ek_i and runs the encryption algorithm on m and replies with the ciphertext $\mathsf{E}_{ek_i}(m)$ and its header hdr. The random coins or nonce r is part of the query input to $\mathcal{O}_E$.

The encryption oracle is needed in the security model since the encryption algorithm uses private keys.

To define security (more specifically, indistinguishability) against chosen plaintext attacks (IND-CPA), we use the following game played between a challenger and an adversary, assuming there is a set U of n source nodes. If no PPT adversary, even in collusion with at most t compromised nodes, can win the game with non-negligible advantage (as defined below), we say the CDA scheme is t-secure. The adversary is allowed to freely choose parameters n and t.

DEFINITION 3. *A CDA scheme is t-secure (indistinguishable) against adaptive chosen plaintext attacks if the advantage of winning the following game is negligible in the security parameter λ for all PPT adversaries.*

Collusion Choice. The adversary chooses to corrupt t source nodes. Denote the set of these t corrupted nodes and the set of their identities by S' and I' respectively.

Setup. The challenger runs the key generation algorithm KG to generate a decryption key dk and n encryption keys $\{ek_i : 1 \le i \le n\}$, and gives the subset of t encryption keys $\{ek_j : s_j \in S'\}$ to the adversary but keeps the decryption key dk and the other $(n-t)$ encryption keys $\{ek_j : s_j \in U \backslash S'\}$.

Query 1. The adversary can issue to the challenger one type of queries:

- *Encryption Query* $\langle i_j, m_j, r_j\rangle$. The challenger responds with $\mathsf{E}_{e_{i_j}}(m_j)$ using random coins r_j. The adversary is allowed to choose and submit his choices of random coins for encryption queries.

Challenge. Once the adversary decides that the first query phase is over, it selects a subset S of d source nodes (whose identities are in the set I) such that $|S \backslash S'| > 0$, and outputs two different sets of plaintexts $M_0 = \{m_{0k} : k \in I\}$ and $M_1 = \{m_{1k} : k \in I\}$ to be challenged. The only constraint is that the two resulting plaintext aggregates x_0 and x_1 are not equal where $x_0 = f(\ldots, m_{0k}, \ldots)$ and $x_1 = f(\ldots, m_{1k}, \ldots)$.

The challenger flips a coin $b \in \{0, 1\}$ to select between x_0 and x_1. The challenger then encrypts each $m_{bk} \in M_b$ with ek_k and aggregates the resulting ciphertexts in the set $\{\mathsf{E}_{ek_k}(m_{bk}) : k \in I\}$ to form the ciphertext C of the aggregate, that is, $C = \mathsf{E}_{\{ek_k : k \in I\}}(x_b)$, and gives C to the adversary. The challenger chooses and passes the nonce to the adversary. The global random coins should be chosen different from those used in the Query 1 phase and no query on them should be allowed in the Query 2 phase.

Query 2. The adversary is allowed to make more queries as previously done in Query 1 phase.

Guess. Finally, the adversary outputs a guess $b' \in \{0,1\}$ for b.

Result. The adversary wins the game if $b' = b$. The advantage of the adversary is defined as: $Adv_{\mathcal{A}} = \left|Pr[b' = b] - \frac{1}{2}\right|$.

Note that in CDA what the adversary is interested in is the information about the final aggregate. Consequently, in the above game, the adversary is asked to distinguish between the ciphertexts of two *different* aggregates x_0 and x_1 as the challenge, rather than to distinguish the two sets of plaintexts M_0 and M_1. Allowing the adversary to choose the two sets M_0, M_1 is to give him more flexibility in launching attacks.

Appendix B: Proof of Theorem 1

Proof: For the sake of clarity, we first prove the security of a version without using the hash function h. Then we show why the proof also works for the hashed version. The reduction is based on the indistinguishability property of a pseudorandom function which is stated as follows:

Indistinguishability Property of a Pseudorandom Function.

Assume f is taken from a pseudorandom function. Then for a fixed input argument x and an unknown, randomly picked key K, the following two distributions are computationally indistinguishable provided that polynomially many evaluations of $f_K(\cdot)$ have been queried:

$$\{y = f_K(x) : y\}, \{y \leftarrow \{0,1\}^{\lambda} : y\}.$$

That is, the output $f_K(x)$ is computationally indistinguishable from a randomly picked number from $\{0,1\}^{\lambda}$ to any PPT distinguisher who has knowledge of the input argument x and a set of polynomially many 2-tuples $(x_i, f_K(x_i))$ where $x_i \neq x$. More formally, for any PPT distinguisher $\mathcal{D}$,

$$|Pr[y = f_K(x) : \mathcal{D}(x,y) = 1] - Pr[y \leftarrow \{0,1\}^{\lambda} : \mathcal{D}(x,y) = 1]| < \varepsilon(\lambda)$$

where $\varepsilon(\lambda)$ is a negligible function in λ.

Proof for the Non-hashed Scheme.

Without loss of generality, we prove the security of a modified version of the construction in which each encryption key is uniformly picked from $\{0,1\}^{\lambda}$, compared with keys generated by a pseudorandom function in the actual scheme. We then provide a justification why the inference applies to the actual implementation.

Suppose there exists a PPT adversary D which can break the semantic security of the scheme with non-negligible advantage Adv_D^{CMT}. We show in the following how D can be used to construct an algorithm D' which can distinguish the above distributions with non-negligible advantage. Assume the key K in question is unknown to D'.

Algorithm D'

Setup. Allow the adversary D to choose any $n-1$ sources to corrupt. Randomly pick $n-1$ encryption keys $ek_i \in_R \{0,1\}^{\lambda}$ and pass them to the adversary. Assume node n is uncorrupted. The encryption key for node n is taken to be K, the key of the pseudorandom function D' is being challenged with.

Query. Upon receiving an encryption query $\langle i_j, m_j, r_j \rangle$ with nonce r_j, return $c_j = (f_{ek_{i_j}}(r_j) + m_j) \bmod M$ if $i_j \neq n$. Otherwise, pass r_j to query the pseudorandom function to get back $f_K(r_j)$ and reply with $c_j = (f_K(r_j) + m_j) \bmod M$.

Challenge. In the challenge phase, receive from D two sets of plaintext messages $M_0 = \{m_{01}, m_{02}, \ldots, m_{0n}\}$ and $M_1 = \{m_{11}, m_{12}, \ldots, m_{1n}\}$.

Randomly pick a number w and output it to the pseudorandom function challenger to ask for a challenge. Note w is the nonce used for CDA encryption in the challenge for D. The pseudorandom function challenger flips a coin $b \in \{0,1\}$ and returns t_b, which is $f_K(w)$ when $b = 0$ and randomly picked from $\{0,1\}^\lambda$ when $b = 1$. These two cases corresponds to the two distributions discussed above.

Randomly flip a coin $d \in \{0,1\}$, and return the challenge ciphertext c_d to D where $c_d = \sum_{i=1}^{n} m_{di} + \sum_{i=1}^{n-1} f_{ek_i}(w) + t_b$.

Guess. D returns its guess b'. Return b'' which is 0 when $b' = d$ and 1 otherwise.

Obviously, if D is PPT, then D' is also PPT. Denoting the expression $\sum_{i=1}^{n} m_{di} + \sum_{i=1}^{n-1} f_{ek_i}(w)$ by X_d, the challenge passed to D can be expressed as $c_d = X_d + t_b$. When $b = 0$, $t_b = f_K(w)$; when $b = 1$, t_b is a randomly picked number from $\{0,1\}^\lambda$. In the following discussion, we denote the output of D on input c_d by $D(c_d)$. The probability of success for D' to distinguish between $f_K(w)$ and a random number is:

$$\begin{aligned}
Pr_{D'}^{PRF}[Success] &= Pr[b'' = b] \\
&= \tfrac{1}{2}\{Pr[b'' = 0 | b = 0] + Pr[b'' = 1 | b = 1]\} \\
&= \tfrac{1}{4}\{Pr[b'' = 0 | b = 0, d = 0] + Pr[b'' = 0 | b = 0, d = 1] \\
&\quad + Pr[b'' = 1 | b = 1, d = 0] + Pr[b'' = 1 | b = 1, d = 1]\} \\
&= \tfrac{1}{4}\{Pr[D(t_0 + X_0) = 0] + Pr[D(t_0 + X_1) = 1] \\
&\quad + Pr[D(t_1 + X_0) = 0] + Pr[D(t_1 + X_1) = 1]\} \\
&= \tfrac{1}{4}\{Pr[D(t_0 + X_0) = 0] + Pr[D(t_0 + X_1) = 1] \\
&\quad + 1 - Pr[D(t_1 + X_0) = 1] + Pr[D(t_1 + X_1) = 1]\} \\
&= \tfrac{1}{4}\{2Pr_D^{CMT}[Success] + 1 - (Pr[D(t_1 + X_0) = 1] - Pr[D(t_1 + X_1) = 1])\}.
\end{aligned}$$

Note that $t_0 + X_0$ and $t_0 + X_1$ are valid ciphertexts for the two challenges plaintext sets M_0 and M_1 respectively. In the last step, we make use of the fact that the probability of success for D to break the semantic security of the scheme is given by:

$$Pr_D^{CMT}[Success] = \tfrac{1}{2} Pr[D(t_0 + X_0) = 0] + \tfrac{1}{2} Pr[D(t_0 + X_1) = 1].$$

Rearranging terms, we have

$$4Pr_{D'}^{PRF}[Success] + Pr[D(t_1 + X_0) = 1] - Pr[D(t_1 + X_1) = 1] = 2Pr_D^{CMT}[Success] + 1$$

$$\begin{aligned}
&4(Pr_{D'}^{PRF}[Success] - \tfrac{1}{2}) + Pr[D(t_1 + X_0) = 1] \\
&\quad - Pr[D(t_1 + X_1) = 1]
\end{aligned} = 2(Pr_D^{CMT}[Success] - \tfrac{1}{2}).$$

Taking absolute value on both sides and substitute $Adv_{D'}^{PRF} = |Pr_{D'}^{PRF}[Success] - \frac{1}{2}|$ and $Adv_D^{CMT} = |Pr_D^{CMT}[Success] - \frac{1}{2}|$, we have

$$2Adv_{D'}^{PRF} + \frac{1}{2} |Pr[D(t_1 + X_0) = 1] - Pr[D(t_1 + X_1) = 1]| \geq Adv_D^{CMT}.$$

Since t_1 is a randomly picked number, $\{t_1 + X_0\}$ and $\{t_1 + X_1\}$ are identically distributed. That is, for any PPT algorithm D, $Pr[D(t_1 + X_0) = 1] = Pr[D(t_1 + X_1) = 1]$. Hence,

$$2Adv_{D'}^{PRF}(\lambda) \geq Adv_{D}^{CMT}(\lambda).$$

Note also that:

$$\big|\, Pr[x \leftarrow \{0,1\}^{\lambda}; y = f_K(x) : D'(y) = 1] - Pr[y \leftarrow \{0,1\}^{\lambda} : D'(y) = 1] \,\big| > 2Adv_{D'}^{PRF}(\lambda).$$

If Adv_{D}^{CMT} is non-negligible in λ, then so is $Adv_{D'}^{PRF}$. As a result, if D can break the semantic security of the scheme with non-negligible advantage, D' could distinguish between the output of pseudorandom function f and a random number. Equivalently, $|Pr[x \leftarrow \{0,1\}^{\lambda}; y = f_K(x) : D'(y) = 1] - Pr[y \leftarrow \{0,1\}^{\lambda} : D'(y) = 1]|$ is non-negligible (a contradiction to the indistinguishability property of a pseudorandom function).

The above security argument applies to the actual implementation since the view of the adversary D in the above simulation is in essence the same as that in the actual scheme. For each one of the $n-1$ corrupted node, the encryption key is $f_{K'}(i)$ ($1 \leq i \leq n-1$) for some randomly picked master key K'. By the property of pseudorandom function, $f_{K'}(i)$ is indistinguishable from a randomly picked key (as used in the above simulation game) for all PPT distinguisher algorithms. For the uncorrupted node, its output for encryption is now $f_{f_{K'}(n)}(x)$ instead of $f_K(x)$ (with randomly picked K) as used in the above simulation game. It can be shown by a contrapositive argument that, for fixed n, the two distributions are computationally indistinguishable, that is,

$$\left\{ K' \leftarrow \{0,1\}^{\lambda}; x \leftarrow \{0,1\}^{\lambda} : f_{f_{K'}(n)}(x) \right\} \stackrel{c}{\equiv} \left\{ K \leftarrow \{0,1\}^{\lambda}; x \leftarrow \{0,1\}^{\lambda} : f_K(x) \right\}.$$

The argument is as follows: Assume f is a pseudorandom function. That is, $A = \{K' \leftarrow \{0,1\}^{\lambda} : f_{K'}(n)\}$ is indistinguishable from $B = \{K \leftarrow \{0,1\}^{\lambda} : K\}$ for all PPT distinguishers. If there exists a PPT distinguisher D which can distinguish between $X = \{K' \leftarrow \{0,1\}^{\lambda}; x \leftarrow \{0,1\}^{\lambda} : f_{f_{K'}(n)}(x)\}$ and $Y = \{K \leftarrow \{0,1\}^{\lambda}; x \leftarrow \{0,1\}^{\lambda} : f_K(x)\}$, we can use D to distinguish between A and B. The idea is when we receive a challenge s which could be from A or B, we send $f_s(x)$ as a challenge for D. If s belongs to A, $f_s(x)$ belongs to X, and if s belongs to B, $f_s(x)$ belongs to Y. We could thus distinguish X from Y (a contradiction).

Security of the Hashed Version.

Only a few modifications to the security proof above are needed in order to prove the security of the hashed variant.

First, in the algorithm D', all ciphertexts are now generated using the hashed values of the pseudorandom function outputs or replies from the challenger of D'. With such changes, we now denote the expression $\sum_{i=1}^{n} m_{di} + \sum_{i=1}^{n-1} h(f_{ek_i}(w))$ by X_d. Of course, the modulus size would be l instead of λ.

Second, the challenge passed to D would be: $c_d = X_d + h(t_b)$. Then the derivation for the advantage expressions is essentially the same as that for the non-hashed scheme.

Third, the security proof of the non-hashed scheme relies on the fact that $\{t_1 \leftarrow \{0,1\}^{\lambda} : t_1 + X_0\}$ and $\{t_1 \leftarrow \{0,1\}^{\lambda} : t_1 + X_1\}$ are identical distribution. On the contrary, to prove the security of hashed scheme, we need the following distributions to be identical:

$$\{t_1 \leftarrow \{0,1\}^{\lambda} : h(t_1) + X_0\}, \{t_1 \leftarrow \{0,1\}^{\lambda} : h(t_1) + X_1\}.$$

If h fulfills the requirement mentioned above, then $\{t_1 \leftarrow \{0,1\}^\lambda : h(t_1)\}$ is the uniform distribution over $\{0,1\}^l$. Consequently, the above two distributions are identical. This thus concludes the proof that hashed scheme is semantically secure.

□

Annexe C

Le protocole Orangina

Cette annexe contient l'article "Shake Them Up !" qui a été présenté à la conférence *ACM/Usenix International Conference on Mobile Systems, Applications, and Services (Mobisys)*, en 2005.

Il présente un nouveau protocole d'échange de clés entre deux appareils. Ce protocole n'utilise aucune fonction cryptographique et est, donc, très bien adapté aux capteurs.

Shake Them Up!

A movement-based pairing protocol for CPU-constrained devices

Claude Castelluccia
INRIA and University of California, Irvine
claude.castelluccia@inria.fr

Pars Mutaf
INRIA
pars.mutaf@inria.fr

Abstract

This paper presents a new pairing protocol that allows two CPU-constrained wireless devices Alice and Bob to establish a shared secret at a very low cost. To our knowledge, this is the first software pairing scheme that does not rely on expensive public-key cryptography, out-of-band channels (such as a keyboard or a display) or specific hardware, making it inexpensive and suitable for CPU-constrained devices such as sensors.

In the described protocol, Alice can send the secret bit 1 to Bob by broadcasting an (empty) packet with the source field set to Alice. Similarly, Alice can send the secret bit 0 to Bob by broadcasting an (empty) packet with the source field set to Bob. Only Bob can identify the real source of the packet (since it did not send it, the source is Alice), and can recover the secret bit (1 if the source is set to Alice or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by Alice or Bob. By randomly generating n such packets Alice and Bob can agree on an n-bit secret key.

Our scheme requires that the devices being paired, Alice and Bob, are shaken during the key exchange protocol. This is to guarantee that an eavesdropper cannot identify the packets sent by Alice from those sent by Bob using data from the RSSI (Received Signal Strength Indicator) registers available in commercial wireless cards. The proposed protocol works with off-the-shelf 802.11 wireless cards and is secure against eavesdropping attacks that use power analysis. It requires, however, some firmware changes to protect against attacks that attempt to identify the source of packets from their transmission frequency.

1 Introduction

The current trend in consumer electronics is to embed a short-range wireless transmitter and a microprocessor in almost everything. The main motivation is to facilitate communication and cooperation amongst wireless devices in order to reduce their size/cost and increase their functionality. In this context, each device can be seen as a peripheral of the others. For example, a user can use the display and the keyboard of a PDA to access his cellular phone or personal server [21]. Similarly, he can use a cellular phone or PDA to retrieve temperature data sensed by a local sensor [19].

The main security challenge is to securely associate the devices together. For example, when a device receives data from a sensor, it needs to make sure that the data is received from the sensor it has selected and not from an impostor. Furthermore, integrity and privacy are often very important too.

The process of securely associating two wireless devices is often referred to as *pairing*. This process allows two devices, communicating over a short-range radio, to exchange a secret key. This key can then be used to authenticate or encrypt subsequent communication. It is important to notice that the key exchanged in a pairing protocol does not need to be authenticated since the identities (often provided by certificates) do not matter in this context. A user who is pairing two devices together only needs assurance that a key was exchanged between the devices he/she has selected (for example, the two devices he/she is holding in his/her hands).

In summary, a pairing protocol is composed of two separate sub-protocols:

1. *Key exchange* sub-protocol: this protocol is run between the two wireless devices and results in a secret key shared between the two devices.

2. *Pairing validation* sub-protocol: this protocol is executed between the two wireless devices and the user. Its goal is to guarantee (with some large enough probability) to the user that a key was exchanged between the two devices he/she actually wished to pair.

Motivations and design constraints: The motivation of this work is to design a pairing protocol for CPU-constrained devices, such as sensors. Designing pairing protocols for such environment is very challenging because sensors have limited CPU and memory. Also, because of their low costs, most of them cannot rely on tamper resistant components. The consequence of the limited computing and storage capabilities is that modular arithmetic is difficult and therefore, asymmetric cryptography cannot be used. In particular, standard Diffie-Hellman (DH)[6] key exchange protocols are excluded. Even low exponent RSA[18] techniques that allow encryption cost to be minimized are prohibitive when sensors are involved. Our goal is to design a pairing protocol that meets these constraints.

More specifically, we aim at designing a protocol that does **not** use public key cryptography and does not rely on some preconfigured information. Furthermore, the designed protocol must not increase the complexity and the cost of the sensors by requiring additional hardware (a display, an I/O interface or an out-of-band channel, such as an infrared one). Finally, it should not require exotic wireless technologies, but instead work with current wireless networking standards such as 802.11 or 802.15.4 (an emerging Wireless PAN technology, designed for low power sensors). The proposed protocol must be secure against passive and active attacks. In other words, it must not allow active or passive attackers to learn the key exchanged between two paired devices. It must provide protection against Man-in-the-Middle (MitM) attacks that attempt to impersonate one or both of the devices during key agreement. It must also provide some protection against Denial-of-Service (DoS) attacks, i.e. prevent attackers from disrupting the pairing protocol and exhausting the devices' resources, such as their battery.

Contributions: We present a novel secure pairing technique based on a key agreement protocol that does not depend on CPU-intensive operations. Two CPU-constrained wireless devices A and B can establish a shared secret over an anonymous broadcast channel. An anonymous channel is a channel on which an eavesdropper can read the packets that are exchanged but is unable to identify the source. Using such a channel, A can send the secret bit 1 (resp. 0) to B by broadcasting an (empty) packet with the source field set to A (resp. B). Only B can identify the real source of the packet (since it did not send it, the source is A), and can recover the secret bit (1 if the source is set to A or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by A or B. By randomly generating n such packets A and B can agree on an n-bit secret key.

The protocol is secure if and only if the packets of A and B cannot be distinguished by an eavesdropper. On a wireless channel, this property is difficult to achieve through protocol design since an eavesdropper can measure the signal strength of each packet and may be able to determine the real source of each packet. Therefore, we propose that during key agreement, the user(s) be very close to each other and shake their devices (i.e. randomly turn and move one around the other) in order to randomize the reception power of their packets by a potential eavesdropper and make power analysis very difficult.

Organization: The paper is structured as follows: Section 2 presents the related work. Section 3 presents the basic ideas of our scheme. Section 4 describes our proposal in detail. Section 5 presents experimental results and analysis. Section 6 presents a discussion. Finally, Section 7 concludes the paper.

2 Related work

2.1 Secure pairing

The problem of secure pairing of wireless devices has been tackled by several researchers. The proposed *key-exchange* solutions can be classified into the four main categories described in this section.

All of the approaches that we review below require some additional mechanism for *pairing validation* (except the last two ones). The only solution proposed in the literature so far is to provide the user with some evidence that both devices computed the same secret key. For example, the devices can both display a hash of the secret key [7]. These solutions are not always practical since they require devices with a display and/or a keyboard.

2.1.1 Public-key cryptography based solutions

These solutions rely on public-key based key exchange protocols such as Diffie-Hellman or RSA [7, 10, 11]. In Diffie-Hellman based schemes, devices exchange their Diffie-Hellman components and derive a key from them. In RSA-based schemes, one of the devices selects a secret key and encrypts it under the other device's public key. The main problem of these solutions is performance. They require that devices perform CPU-intensive operations such as exponentiation, which are prohibitive for CPU-constrained devices.

2.1.2 PIN-based schemes

In Bluetooth, two wireless devices derive a shared key from a public random value, the addresses of each device and a secret PIN number. The PIN number is provided to each device by the user via an out-of-band channel, such

as a keyboard. While this solution is computationally efficient, it requires that *both* devices be equipped with some kind of physical user interface. As a result, this solution cannot be used to pair devices lacking physical interfaces, such as sensors.

2.1.3 Physical contact or imprinting

In [19], Stajano and Anderson propose a solution, called *imprinting*, based on physical contact. Two devices get paired by linking them together with an electrical contact that transfers the bits of a shared secret. No cryptography is involved, since the secret is transmitted in plain-text. Furthermore, the key validation phase is not necessary since there is no ambiguity about the devices that are involved in the binding (i.e. MitM attacks are impossible).

While this solution is interesting, it requires each device to have some additional hardware to perform the electrical contact. Similarly, it might be possible to transmit a secret key through an infrared channel to a nearby node. Infrared transmissions require absolute line-of-sight links, making it more difficult for third-party interception. Nevertheless, in both cases, i.e. physical contact or infrared transmission, the complexity and the cost of the devices would increase[1]. This violates one of our design requirements, that the pairing protocol should not require extra equipment. We wish to achieve key agreement through the actual communication channel.

2.1.4 Using a Faraday cage

Alternatively, the two devices can be put in a *Faraday cage* where the secret key is transmitted in plain-text. A Faraday cage is an electrical apparatus designed to prevent the passage of electromagnetic waves, either containing them in or excluding them from its interior space. Consequently, an eavesdropper could not hear the secret key. An idealized Faraday cage is a hollow electrical conductor such as an empty sphere or box. Practical Faraday cages can be made of a conducting mesh instead of a solid conductor. However, this reduces the cage's effectiveness as an RF shield. In our case, the paired devices are small and hence can be enclosed in a conducting box. However, this solution is probably impractical for the general case; a conducting box is not always available. Users cannot possibly foresee when and where they will need secure pairing (and clearly we cannot recommend a user to carry a metal box with her all the time).

Similarly to a Faraday cage, a cable could be used for secret key transmission, instead of wireless link. However, users typically do not have cables available when they need to communicate, and requiring it will be a considerable impediment to secure communication.

One of our design requirements is to develop a protocol that does not require extra hardware/equipment. Solutions based on using a Faraday cage or cable clearly violate this constraint.

2.2 Other shaking-based schemes

As we will describe in detail, "Shake Them Up!" security depends on *shaking* two devices. "Smart-Its Friends"[12], although not related to key agreement nor security, is based on a similar user-device interaction. The authors propose that sensors be equipped with a two-axis accelerometer. When a user takes two devices in one hand and shakes them, the devices generate and broadcast similar movement data. If the difference is below a specified threshold, then the two devices recognize each other as friends and a dedicated connection is established between them. In another related work, "Are you with me?"[15], the authors propose using accelerometers to determine if two devices are carried by the same person.

In "Shake Them Up!", the shaking process has a completely different role (and no accelerometers are used in this paper). Devices are shaken/rotated for randomizing the signal power of the messages received by a potential eavesdropper.

3 Basic ideas

This section describes the main ideas of our scheme. We first describe how two devices, communicating over an anonymous channel, can exchange a secret key without CPU-expensive computation. We then define formally what we mean by "anonymous channels" and describe how they can be implemented in practice.

3.1 Pairing over anonymous channels

This section describes a new technique that allows two parties to securely exchange a secret over an anonymous channel while preventing eavesdroppers from determining its value (or actually any of its bits). By anonymous channel, we mean a broadcast channel that hides the origin of the messages. On an anonymous channel, a passive wiretapper can read the messages that are broadcast, but is unable to identify the source. Anonymous channels require a property that we call *Source Indistinguishability*. This property is defined and discussed in Section 3.2.

Our key exchange protocol was inspired from the protocol proposed by Alpern and Schneider in [3]. In this paper, the author presents a protocol that allows two parties to agree on a secret key on channels for which an eavesdropper cannot tell "who" broadcasts each message. The technique is called "Key exchange using keyless cryptography", or "Keyless key agreement".

For users Alice (A) and Bob (B) to agree on a n-bit key $K_{AB}[n]$, each first chooses its own random 2n-bit string:

$$R_A[1], R_A[2], ..., R_A[2n]$$

$$R_B[1], R_B[2], ..., R_B[2n]$$

User A then broadcasts $2n$ anonymous messages (without sender identifier), one for each bit in R_A. Similarly, user B broadcasts $2n$ anonymous messages (without sender identifier), one for each bit in R_B. The secret key is then defined by the bits $R_A[j]$ sent by A such that $R_A[j] \neq R_B[j]$. Note that there are on the average n such bits. The salient property of the protocol is that the message content is not hidden. All messages are accessible to potential eavesdroppers, which however cannot determine the origin of each message. As a result, they are unable to identify the packets sent by A and therefore identify the correct bits. Since A knows her bits, she can easily identify the bits sent by B. Similarly since B knows his bits, he can easily identify the bits sent by A. Note that the packets transmitted by A and by B must be interleaved. Otherwise it might be easy for an eavesdropper to identify the bits sent by the same source from a timing analysis. $R_A[1]$ and $R_B[1]$ should be sent first, followed by $R_A[2]$ and $R_B[2]$, the transmission order of each pair being randomized, and so on.

The protocol presented by Alpern and Schneider requires the broadcast of $4n$ messages for A and B to agree on an n-bit secret key. We propose an optimization that reduces the number of broadcast messages to n. The overview of our protocol is the following (a more detailed description is presented in Section 4):

1. A selects n/2 random bits $R_A[1], R_A[2], ..., R_A[n/2]$
2. B selects n/2 random bits $R_B[1], R_B[2], ..., R_B[n/2]$
3. A builds n/2 messages $m_A[1], m_A[2], ..., m_A[n/2]$, where the source identifier of $m_A[j]$ is either set to A if $R_A[j] = 1$ or set to B if $R_A[j] = 0$.
4. B builds n/2 messages $m_B[1], m_B[2], ..., m_B[n/2]$, where the source identifier of $m_B[j]$ is either set to B if $R_B[j] = 1$ or set to A if $R_B[j] = 0$.

- A and B send their messages synchronously but in a random order. In other words, the first messages of A and B are sent (in a random order), followed by the second messages (in a random order), and so on. In total, n messages are sent.
- For each message that A (resp. B) receives, it checks whether the source identifier is set correctly (note that only A and B can perform this verification) and sets $K_{AB}[k]$ to 1 if the source is correct or to 0 otherwise.

3.2 Source indistinguishability: definition and requirements

The described key exchange protocol requires the *source indistinguishability* property. In other words, if two parties, A and B, run the previously described key exchange protocol, an eavesdropper should not be able to distinguish the packets sent by A from the packets sent by B. Failing to achieve this property actually leads to an insecure protocol, since the eavesdropper could then recover some (if not all) bits of the exchanged key.

This notion of source indistinguishability is very similar to the notion of ciphertext indistinguishability in encryption schemes [8]. The basic idea behind indistinguishability of an encryption scheme is to consider an adversary (not in possession of the secret key) who chooses two messages, m_1 and m_2, of the same length. Then one of the messages is encrypted and the ciphertext is given to the adversary. The encrypted scheme is considered secure if the adversary cannot tell which of the two messages was encrypted.

We define *source indistinguishability* in a similar way as follows: a communication scheme between two parties A and B is *source indistinguishable* if for a given packet P, emitted by A or B, an eavesdropper cannot tell whether the packet was sent by A or B. More formally, the difference between the probability that the packet was sent by A and the probability that the packet was sent by B should be very small:

$$Pr[source(P) = A] - Pr[source(P) = B] < \epsilon$$

In practice, source indistinguishability requires the communication to be *temporally* and *spatially* indistinguishable. In the following sections, we discuss these requirements in detail [2].

3.2.1 Temporal indistinguishability

Given two parties A and B communicating together, an eavesdropper should not be able, using *timing analysis*, to identify the packets emitted from A from those emitted from B with a probability larger than $1/2$. Furthermore, the eavesdropper should not be able to group packets emitted by the same source.

It is clear from the previous definition that a communication system that uses a TDMA (Time Division Multiplexing Access) based MAC (Medium Access Control) protocol cannot provide temporal indistinguishability. In a TDMA-based system, each terminal is given one or several time slots and can transmit only during one of

its slots. As a result, it is very easy for any eavesdropper to identify the packets transmitted by a source or at least identify the packets sent by the same source.

Random access MAC protocols, such as CSMA (Carrier Sense Multiple Access) are more appropriate. CSMA protocols such as Ethernet or wireless Ethernet, multiple nodes are allowed to use the same channel in a random fashion. Before transmitting data a node listens to the channel. If the channel is busy then it waits for a random time and then listens again. If the channel is not busy, then it transmits its packet. In CSMA, collisions may happen when two terminals transmit simultaneously. CSMA/CD (Collision Detection) enables devices to detect a collision. After detecting a collision, a device waits a random time period and then attempts to re-transmit its message. With a CSMA-based system, the order of the packets sent by the different users can easily be randomized. This feature is crucial for the security of our approach. In this case, it will be very difficult for an eavesdropper to use timing information to identify the source.

3.2.2 Spatial indistinguishability

Given two parties A and B communicating together, an eavesdropper should not be able, using *spatial analysis* (or signal strength analysis), to distinguish the packets emitted by A from those emitted by B with a probability larger than $1/2$. In other words, the eavesdropper should not be able to detect the packets' source from their reception power.

This property is very difficult to achieve in practice since waves attenuate according to the *free space propagation* law and the eavesdropper can easily identify the location of the transmitter from the reception power of a received packet, i.e. from *power analysis*. More specifically, according to free space propagation law, the reception power S_r of a packet transmitted with power S_t by a transmitter that is located at a distance d is defined as: $S_r = \frac{S_t\ G_t\ G_r\ K}{d^2}$, where G_t is the antenna gain of the transmitter, G_r is the antenna gain of the receiver and K is a constant that depends on the signal frequency (or wavelength). If the receiver knows S_t and the gain, it can easily estimate d. An eavesdropper listening to a communication between two parties A and B can also use the reception power to identify the source of the packets with a probability larger than $1/2$.

Let's assume that A and B use the same type of antenna (i.e. they have the same gain) and let's define $k = G_t\ G_r\ K$. If A and B transmit their packets with a power uniformly distributed between $[S_t; S_t + \delta]$ then A receives the packets from B with a power uniformly distributed between $[\frac{k\ S_t}{d^2}; \frac{k\ (S_t+\delta)}{d^2}]$. Similarly, B receives the packets from A with a power uniformly distributed between $[\frac{k\ S_t}{d^2}; \frac{k\ (S_t+\delta)}{d^2}]$. If the eavesdropper is listening with a large antenna (i.e. G'_r is large s.t. $k' = G_t\ G'_r\ K > k$) and it is closer to A than to B (i.e. $d_A \leq d_B$), then:

1. the power of A's packets received by the eavesdropper is uniformly distributed between $[\frac{k'\ S_t}{d_A^2}; \frac{k'\ (S_t+\delta)}{d_A^2}]$ and,
2. the power of B's packets received by the eavesdropper is uniformly distributed between $[\frac{k'\ S_t}{d_B^2}; \frac{k'\ (S_t+\delta)}{d_B^2}]$.

Therefore the eavesdropper can identify all the packets received with a power between $[\frac{k'\ S_t}{d_B^2}; \frac{k'\ S_t}{d_A^2}]$ as belonging to B and all the packets received with power between $[\frac{k'\ (S_t+\delta)}{d_B^2}; \frac{k'\ (S_t+\delta)}{d_A^2}]$ as belonging to A. The scheme can only be secure if one of the two following conditions is met:

1. *Condition 1*: $d_A = d_B$. The scheme is secure because the power of the packets sent by A is statistically indistinguishable from the power of the packets sent by B. If $d_A \neq d_B$, the eavesdropper can identify some of the bits of the secret key. The number of bits that can be identified depends of the values of d_A and d_B. If $d_B > (S_t + \frac{\delta}{S_t})^{1/2} d_A$, the eavesdropper can guess the source of all the packets exchanged between A and B and therefore all bits of the secret key. If $d_A < d_B < (S_t + \frac{\delta}{S_t})^{1/2} d_A$, the eavesdropper can guess the source of some percentage of the packets. This percentage depends on the difference between d_A and d_B. Note that if the eavesdropper can monitor at several locations, and receive the same packets with different reception powers, it will be even easier for her to identify the source of the packets.
2. *Condition 2*: A and B move during the pairing phase such that d_A and d_B (and therefore their respective powers) are statistically indistinguishable.

4 Movement-based pairing

This section describes a new protocol that can be used to pair two devices A and B securely and without using expensive public-key cryptographic protocols. Our key agreement protocol is based on the combination of the two following: we first optimize Alpern and Schneider's keyless key agreement protocol and adapt it to an ad hoc configuration. Secondly, we show that the protocol can be secured against power analysis by shaking the two devices around each other. We show that it is secure against DoS (Denial-of-Service) and MitM (Man-in-the-Middle) attacks.

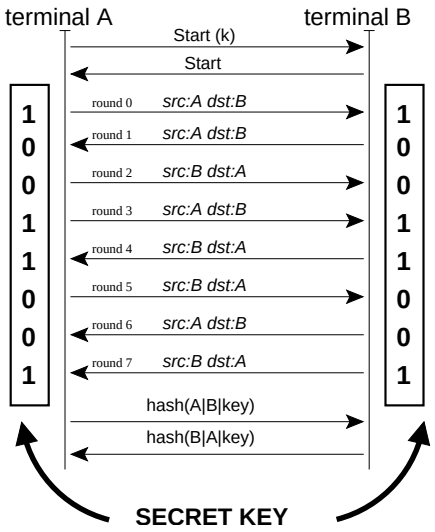


Figure 1: Key agreement protocol for movement-based pairing.

4.1 Key agreement

In our pairing protocol, key agreement is performed using the protocol described in Figure 1. This protocol is an optimization of the approach proposed by Alpern and Schneider [3]. The number of messages per secret bit is reduced (1 instead of 4), making the protocol more energy efficient.

The protocol starts by a START message transmitted by one of the two parties, either A or B. This message contains the value k which is the size of the key, and the address of the packet's source. Upon reception of this message, the other party replies with another START message that contains its address. This exchange allows each device to learn the other party's address and the size of the desired shared key. It should be triggered by the user, for example, pushing a button on both devices.

At each round j, either A or B (with equal probability) broadcasts an empty packet at time t_j, where t_j is randomly selected in the interval $[jT; (j+1)T]$, and T is a constant. Secret bits are represented by the correct or inversed placement of source and destination addresses. If the sender and recipient addresses are correct, the terminals A and B presume a secret bit TRUE (1), otherwise they presume FALSE (0). For example, in Figure 1, the first message is sent by A. The source address is set to A and the destination address is set to B. Since the packet was actually sent by A, the resulting bit (the first bit) of the secret key is then set to 1 by A and B. Note that an eavesdropper cannot identify the real source of the packet and therefore cannot retrieve the value of the exchanged secret bit. Each packet identifies one bit of the secret key. At the end of the k rounds, A and B share a k-bit long secret key. Therefore, if a 60-bit long key is required, the protocol should contain 60 rounds, i.e. 60 packets.

The protocol is terminated by two messages, that are used to validate the exchanged key. The message sent by A contains the value $a = hash(A|B|key)$, where key is the exchanged key. The message sent by B contains the value $b = hash(B|A|key)$. The order of these two messages is also random. When B receives the value a, it can verify that A has the same key. Similarly, A can verify, upon reception of b, that B computed the correct key.

4.2 Achieving spatial indistinguishability: Shake them up!

As described in Section 4, the previous scheme is secure only if the source of the packets are indistinguishable.

Time indistinguishability is provided by randomizing the order of transmission of packets sent by A and B, as described in the previous section. An eavesdropper can therefore not guess who is going to transmit next. Also, we are using CSMA-based wireless systems, such as 802.11, to guarantee that the access to the channel is also random and does not reveal any information about the source.

As explained previously, *spatial indistinguishability* is more difficult to achieve. We propose to achieve this property with user assistance. The user(s) should shake (i.e. move and turn) the devices during key agreement in order to equalize the average signal strength of the two devices measured by a potential eavesdropper.

The required movements depend on the type of antennas used. For truly omni-directional antennas, antenna orientation will not reveal any signal strength difference between two devices. In these cases, it will be sufficient to take both devices and turn them, one around the other, in order to equalize the effect of distance (between each terminal and the eavesdropper) on the signal strength measurements performed by an eavesdropper. If the antennas are not truly omni-directional, randomizing the distance will not be enough to achieve spatial indistinguishability. Different orientation of the devices may reveal a serious signal level difference. In order to avoid this problem, during key agreement the two devices must be randomly turned to different directions (in our experimentations we used commodity 802.11 cards which are not omni-directional. Section 5 contains more details on this issue).

Clearly, in both cases, having small and lightweight devices (e.g. sensors or small PDAs) will reduce the user burden for key agreement. The user can take one device in each of his hands and randomly move them one around the other according to the horizontal and vertical axes. If the devices are very small, he can take both of them in one hand and shake his arm, like he would do with an orange juice bottle.

The security of the proposed scheme depends on the quality of the movement. Users of our scheme should be aware that it is their responsibility and in their best interest to move the devices properly during the pairing phase. Movement-based operations or "protocols" are quite frequent and accepted in our everyday lives. For example, orange juice or shaving cream bottles are universally shaken prior to use. This is now a well-known and quite a natural protocol. Furthermore it is commonly accepted that it is the responsibility and in the interest of the consumer to perform this shaking operation properly.

4.3 Protection against MitM and DoS attacks

4.3.1 Protection against MitM attacks

Defeating a MitM (Man-in-the-Middle) attack requires assurance for a terminal A that a secret key is really being exchanged with the intended terminal B and not an impostor's device. This is the goal of the *Pairing Validation* protocol, as described in Section 1.

In our case, this problem is reduced to the following issue: "how do the two devices reliably determine each other's address in the presence of many other devices?" As explained in Section 2.2, the smart-its friends scheme solves this problem. However this solution may be considered costly and impractical since it requires extra hardware, namely an accelerometer. Certificates could also be used for authenticating the START messages. However, our goal in this paper is to avoid CPU-expensive operations such as exponentiations or signature verifications.

In our case, $proximity$ (and hence high signal level of START messages) is used for authentication. In "Shake Them Up!", the pairing protocol is triggered by the user by, for example, pushing a button on the devices. At this point, the devices will start to generate START messages at a specified rate. The user will bring the two devices near to each other (possibly resulting in antenna contact). The devices will receive each other's START message with a high signal level, starting the "Shake Them Up!" procedure. Experience with 802.11 cards has shown that a very high signal level can be obtained when two cards touch each other, and a distance about 1 or 2 cm quickly results in a much lower signal level. Using a signal level threshold e.g. 0 or 1 dBm, this device association protocol can be implemented. In our case, the higher the specified rate of START messages, the faster the devices can detect each other with high signal level (when the user finds the correct positioning). Let Q be the period of broadcast START messages. While initiating the pairing process, if the user missed a START message (i.e. it was received at a lower signal level than the specified threshold), the user must wait another Q time units. Consequently, Q must be low, e.g. for example 1 second, to allow the user to easily and quickly start the pairing process. Once the two devices obtain each other's address correctly, MitM attacks will be impossible.

A distant impostor may attempt to foil this technique by increasing its transmission range. However this attack is easily detectable since the victim will receive several START messages at a rate higher than Q.

One of the devices, say device B, may be down and an attacker may profit from this situation to impersonate B. This attack can be prevented if each device has a status LED which indicates whether it is active or down. Even a sensor device (with no display) is likely to have such a LED. Furthermore, if the devices A and B are shaken together (i.e. the user holds the two devices in one hand and shakes his hand), A should receive the messages coming from B with constant signal power. Most likely, the signal power of the different messages sent by the attacker will be different (since the attacker is not shaken together with A and therefore does not follow the same mobility pattern). In this scenario, this attack can easily be detected by A.

4.3.2 Protection against DoS attacks

We can differentiate between two kinds of DoS (Denial-of-Service) attacks on a key agreement protocol. In the first one an attacker may exploit the key agreement protocol to force a victim to perform computationally expensive operations, with the goal of draining its battery or preventing it from performing useful work. Unlike public-key cryptography-based schemes, our protocol is not based on CPU-intensive operations and therefore immune against such DoS attacks. Another DoS attack may consist of sabotaging the key agreement, i.e. making it impossible for the two parties to agree on a same secret key.

In our basic scheme, it is easy for an attacker to insert a bogus packet with source address A and destination address B (or vice versa) and perform what we call a *key poisoning* attack. Such a packet inserted by a third party would generate different secret bits at the terminals A and B. The attacker can insert an arbitrary number of bogus packets, and make it impossible for A and B to agree on a secret key.

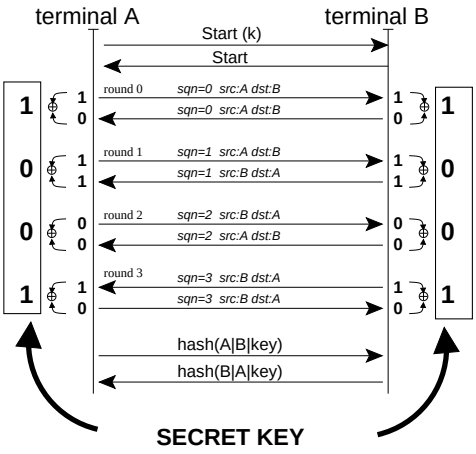


Figure 2: A protocol that resists what we call a *key poisoning DoS attack*. When this protocol is used for key agreement, an active attacker cannot poison (i.e. corrupt) the secret key by inserting bogus messages with the addresses of A and B (see text for details). This protection is obtained at the cost of two messages per secret bit.

The protocol depicted in Figure 2 defeats the key poisoning attack. In this protocol, each secret bit is constructed using one packet from A and another from B, i.e. both terminals contribute to the construction of each secret bit. Each secret bit is given a sequence number (which also corresponds to the round number). In order to generate the secret bit i the two terminals generate a packet with correct or flipped address positions, in random order (i.e. the probability that the first packet i will be transmitted by A is 0.5). The outcome of the two packets i are combined by taking their sum (mod 2), or exclusive OR. The result is the secret bit i.

In order to alterate the secret bit i, an attacking node can insert x packets with the same sequence number. In this case both sides will record $x + 2$ bits with the same sequence number, but only two of them will be the same at both sides. Let $\{a_1, ..., a_{x+2}\}$ and $\{b_1, ..., b_{x+2}\}$ the set of bits (with the same sequence number) that the terminals A and B have recorded. Then we have

$$a_1 \oplus \cdots \oplus a_{x+2} = b_1 \oplus \cdots \oplus b_{x+2}$$

if x is even, and

$$b_1 \oplus \cdots \oplus a_{x+2} \neq b_1 \oplus \cdots \oplus b_{x+2}$$

if x is odd.

Key poisoning can be defeated by taking the sum (mod 2) of the bits with the sequence number i, as usual (except that in normal operation $x = 0$). Note that if the attacker inserted an odd number x of packets, the terminal A must invert the resulting secret bit i. This protocol requires twice as many messages as the basic protocol. However, this protocol is only necessary when the user believes that his devices are under DoS attack. Otherwise, the basic scheme should be used.

This protocol fails, however, when A and B do not receive the same messages. This might happen when some of the messages are lost. We therefore suggest that A and B append to each of their messages a hash of all the previous messages they have seen since the beginning of the protocol. As a result, if one or several messages are lost, A and B can detect it immediately (instead of waiting until the end of the protocol and comparing a hash of the derived keys).

5 Experimentations and analysis

In this section, we analyze, by experimentations, the security of the proposed pairing scheme. More specifically, we show that signal power analysis cannot be used by an attacker to retrieve the key exchanged between two devices that use our pairing protocol. We also evaluate the energy cost of our protocol and show that although it requires several messages, it is much more energy-efficient than a Diffie-Hellman based pairing protocol.

5.1 Setup and methodology

We built a testbed with lightweight laptops equipped with a PCMCIA Lucent IEEE 802.11 Wavelan card operating at 2.457GHz (802.11 channel 10) and 2Mbps bit rate and in ad hoc mode. Our cards support RSSI (Received Signal Strength Indicator) and allow us to visualize and evaluate the power of received packets. Our signal level cryptanalyzer is built upon Linux wireless tools[3], and in particular *iwspy* that allows to get per node link quality. The $iwspy$ command takes as argument a MAC address M, and outputs the received signal and noise levels of packets whose source address is M.

Figure 3 depicts a typical measurement that can be carried out by any user using the $iwpsy$ tool and a simple sampling script. Note that $iwspy$ also outputs the noise level which is about -96dBm in our environment.

The received signal strength depends on at least 4 distinct factors:

1. Transmission power: Packets are transmitted at our cards' default value which is 15 dBm[4].
2. Distance between the source and the signal level analyzer.
3. The relative angle between the source and the signal level analyzer: the cards that we use are not

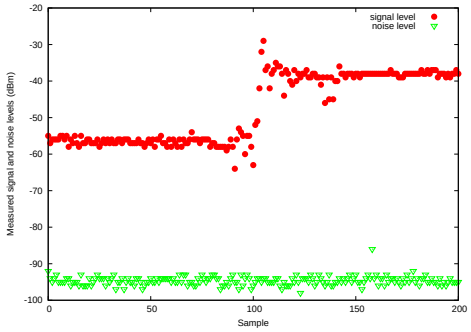


Figure 3: A typical *iwspy* output. In this example, the "iwspied" terminal is stationary while the first ∼100 samples are taken and then it moves to a location that is closer to the signal level analyzer machine. The distance between the two cards is easily determined by signal strength analysis.

omni-directional and the received signal level depends heavily on the relative angle of the two cards.

4. Relative position of the terminals (or, cards): when the two terminals are very close, they may become obstacles for each other. For example, terminal A may be in front of terminal B. In this case, packets from A are received at a higher signal level (even if the above factors had no impact on the signal level difference).

During each experiment, referred as 'scenario', Eve (eavesdropper, or cryptanalyzer) measures the signal strength of the packets sent by Alice (terminal A) and Bob (terminal B) during key agreement. Many different experiments were carried out. We only provide the most representative ones that we consider generic and applicable to almost all situations because they perfectly reflect the points (2) and (3) listed above. Our first scenario, denoted *scenario1*, is illustrated in Figure 4-a. In this scenario Alice and Bob are close to each other (within 0.5 meter) and make two kinds of movements in order to equalize their average signal strength captured by Eve:

- We use commodity wireless Ethernet cards and they are not omnidirectional. Thus, in order to confuse Eve, Alice and Bob must turn their laptop in randomly changing directions (at a reasonable speed). The process requires reasonable effort from the user and takes around 16 seconds for agreeing upon an 80-bit secret key. The details of movement speed and its effect on security will be discussed later.
- Alice and Bob move their devices one around another with a reasonable effort, i.e randomly and at a moderate speed. This helps Alice and Bob to hide their relative distance between their cards and a potential eavesdropper that may be located anywhere.

In scenario1, we consider a pessimistic passive attack where Eve's wireless Ethernet card (the white arrow) is directly oriented to Alice and Bob and situated only 2.2 meters away. This allows Eve to make relatively accurate signal level measurements. In practice, Alice and Bob would probably notice the presence of a third person during key agreement, and look for another place where eavesdroppers cannot approach them. However, in some situations such countermeasures may not be practical. This scenario attempts to capture the cases where the presence of a third person cannot be avoided. Note also that an eavesdropper may have installed hidden signal level cryptanalyzers at strategical points. Thus, the absence of a third person, does not necessarily imply a secure environment. For example, the eavesdropper might be behind a thin wall or partition (e.g. a cubicle wall), and not visible to Alice and Bob. In this scenario Alice and Bob respect the key agreement requirements, hence the key will be secure as we will show below.

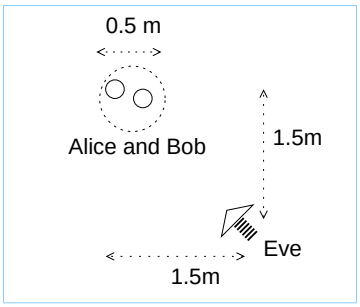
In scenario2 (Figure 4-b), we demonstrate an inappropriate usage of our protocol that we would like to disadvise. In this scenario Alice and Bob are not close to each other. They both move their laptop randomly in every possible direction, but they are always far from each other and their location does not change during key agreement. Eve profits from the distance between Alice and Bob, and directs her card to Alice. Consequently, Alice's packets are received at a higher signal level than that of Bob, rendering the secret key weak.

The scenario3 (Figure 4-c) is even less secure and firmly disadvised. Alice is 4 times closer to Eve than Bob. Eve is located between the two terminals and profits from the situation by directing her wireless Ethernet card to Alice. Although Alice and Bob's cards are perfectly turned in random directions, Eve can easily differentiate between their packets. As a result, the key is extremely weak.

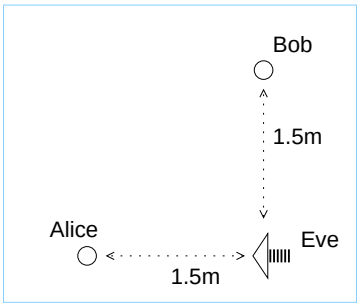
5.2 Signal power analysis

The signal level cryptanalysis results for the above three scenarii are shown in Figure 5.

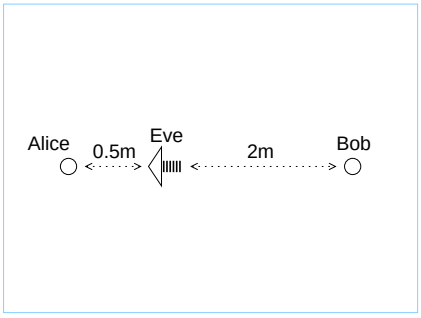
In scenario1, we observe that Alice and Bob's packets are mixed and not easily distinguishable by Eve (the reader may imagine that Eve's vision has only one color regardless of the sender's ID). The only information available to Eve will be the absolute value of signal level difference between two packets captured during



(a) scenario1

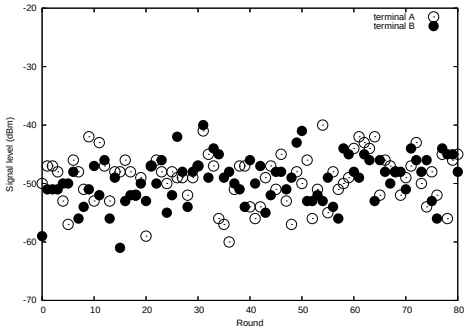


(b) scenario2

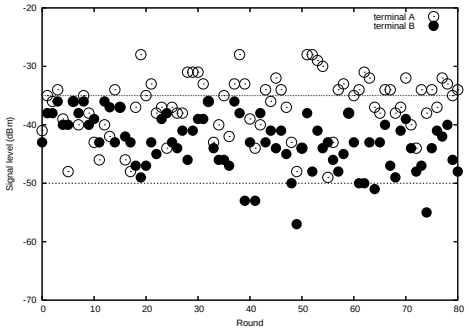


(c) scenario3

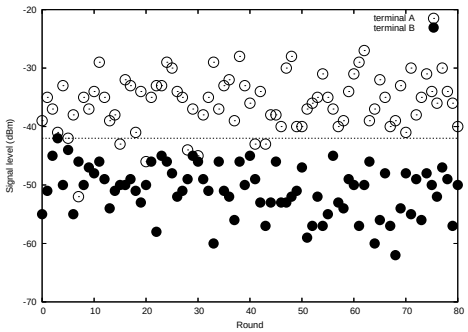
Figure 4: Experimentation scenarii.



(a) scenario1



(b) scenario2



(c) scenario3

Figure 5: Received signal level of terminal A (Alice) and terminal B (Bob) during key agreement. The reader may imagine that Eve's vision has only one color (i.e. all packets are black).

each round (1 packet from Alice, 1 packet from Bob). In Figure 6 we provide a frequency diagram of these signal level differences. It should be noted that, when plotting these histograms we have profited from additional information that is not available to Eve: the sign of the observed differences (i.e. (+) when Alice's packet is received with greater signal level than that of Bob, and (-) otherwise). These histograms were plotted using 1000 samples (i.e. rounds) in order to provide accurate results that correspond to the average case (for a given scenario). For our data collection purposes, Alice and Bob performed the required laptop movements for a much longer time than needed in practice: ~ 3.5 minutes for each experiment (in our experiments Alice and Bob generated 0.2 packets per second, as we will explain later). The histogram that corresponds to scenario1 is centered on $\sim$0 and roughly symmetric. Consequently, we conjecture that Alice and Bob's packets cannot be distinguished using signal strength analysis.

In practice, the results look satisfactory in scenario2. There is no well defined technique (at least to our knowledge at time of writing) that will allow to clearly distinguish Alice's and Bob's packets. Note for example that, above the line -50dBm all packets are Alice's packets. Similarly, below the line -35dBm, we have only Bob's packets. However, unlike the reader, this information is not provided to Eve.

On the other hand, the resulting key is clearly insecure in theory. As shown in Figure 5-b the signal level differences are important: the histogram is centered at 7.34 dBm. Although it is unknown to the attacker, this difference is considerable (making us uncomfortable) and reflects very well the fact that Eve's wireless Ethernet card is directed to Alice. Thus, we base our conclusion on the theoretical security of the protocol and disadvise the type of scenario where Alice and Bob are 'not' close to each other. The security of the secret key in this scenario could be improved by adding more rounds, however this would lead to a very inefficient key agreement. We recommend that Alice and Bob be as close to each other as possible (in addition to turning their devices in random directions).

In the final scenario, scenario3, the situation is clearly worse. The resulting key is not only 'theoretically' breakable as shown by the frequency diagram (centered at 14.92 dBm), but also breakable in practice. Figure 5-c reveals what we call a "break point" which is situated around -41dBm. There is a visible gap at that point where Alice and Bob's packets are clearly separated.

5.3 Energy consumption considerations

In this section, we compare the power consumption of our scheme with the power consumption of a Diffie-Hellman based pairing. For the purpose of this comparison, we assume that the two devices being paired are sensors using TinyOS and that the size of the generated shared key is 72 bits.

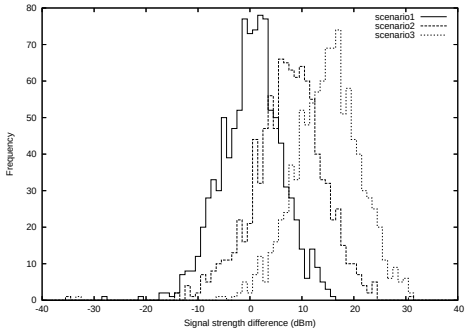


Figure 6: Frequency diagrams for signal level difference. In scenario1, the spatial indistinguishability requirement is satisfied. The histogram is centered on zero and symmetric. Thus, in this paper it is conjectured that an eavesdropper cannot distinguish the source of the packets regarding signal level difference (in scenario1).

With our scheme, each device must receive and send 36 packets. Considering that a TinyOS packet that has a header size of 7 bytes [13], each device must send and receive 2016 bits ($36 \times 8 \times 7$) (as explained in Section 4, in our basic protocol, packets do not have to contain any payload). However transmitting one bit consumes about as much power as executing 800-1000 instructions [9, 13]. Receiving one bit consumes about half as much power as sending one bit. As a result, our protocol consumes as much energy as the execution of about 2.72×10^6 instructions ($2016 \times 900 + 2016 \times 450$).

In comparison, with a Diffie-Hellman based pairing protocol, each device needs to exchange their Diffie-Hellman public component (i.e. g^x, where x is the device's private key). A security equivalent to 72 bits requires to select a modulus of 1024 bits and an exponent of 130 bits [14]. As a result, the device's Diffie-Hellman public component is 1024-bit long. Since the maximum number of payload bits in a TinyOS packet is 232, each device must send (and receive) 5 packets. Therefore, the total number of bits sent and received is 1304 bits: 4 packets containing 232 bits and 1 packet containing 96 bits. This consumes as much energy as the execution of 1.76×10^6 instructions ($1304 \times 900 + 1304 \times 450$). Upon reception of the other party's public component, each device has to exponentiate it with its Diffie-Hellman private key. Exponentiating using the Montgomery algorithm re-

quires $3 \times l \times (l+1) \times (t+1)$ single-precision multiplications, where l is the size of the modulus and t the size of the exponent [16]. With $l = 1024$ and $t = 130$, each device must perform 4.12×10^8 single-precision multiplications. In conclusion, the total power consumed by each device is therefore equivalent to the power consumed by the execution of $1.76 \times 10^6 + 4.12 \times 10^8$ instructions. This cost is about *100 times* larger than the cost of our scheme.

The bandwidth cost of the Diffie-Hellman based solution could be significantly decreased with Elliptic Curve Cryptography. In fact, the security of a 1024-bit Diffie-Hellman key exchange is equivalent to the security of a 135-bit Elliptic Curve Diffie-Hellman (EC-DH) key exchange [14]. Therefore only one packet would be necessary to be exchanged by the two devices. This would reduce the energy cost due to communication by 5. However, as shown in [5], EC-DH key derivation cost is even more expensive than regular DH key derivation. Therefore the total energy cost would still be much higher than the cost of our scheme.

6 Discussion

In this section, we present a discussion of more sophisticated attacks that the "Shake Them Up!" strategy may face.

6.1 RF analysis attacks on reference clocks

How secure is our protocol against a well equipped attacker? An attacker may use more sophisticated equipment, and conduct much more rigorous cryptanalysis. Using an RF test equipment, for example, it is possible to snoop the packets and record their signal shape. By studying the signal shape of each packet, additional information may be discovered and help distinguish one of the device's packets from the other. Although the signal amplitude should not reveal anything more than an RSSI measurement, signal-frequency may reveal a drift between the participants' reference clock implementation.

By current practice, a quartz crystal or crystal clock oscillator stabilizes the carrier and baseband frequencies in an RF transceiver. In order to ensure frequency lock between two devices, and avoid serious phase noise, a tight-stability reference clock is necessary. Nevertheless, a reference clock implementation is never perfect. A random clock drift is generally unavoidable, due to practical difficulties found at the hardware level. Typically, an error of up to $\pm$25ppm (parts per million) is tolerated. This tolerance includes the initial calibration tolerance at $25°$C, frequency changes over operating temperature, power and load fluctuations, and aging[17]. For example, at 2.4GHz carrier frequency, a frequency offset of up to $\frac{2.4\times10^9\times2\times25}{10^6}$ =120kHz would be tolerated. [4] reports 250kHz of central frequency accuracy tolerance. The main reason for clock drift is aging. I.e. the clock drift is mostly stable in short-term (except in case of shock, or abrupt temperature changes), but logarithmically increases in time[20, 1]. A clock drift from ±1 to ±5 ppm/year can be incurred depending on the crystal used [2].

Consequently, during "Shake Them Up!", device A may always transmit at a central frequency f_A while the device B transmits at f_B, where $f_A = f_B + \epsilon$. A third party equipped with an RF analyzer can then retrieve the secret key by correlating the packets with the same central frequency. A frequency shift of several kHz is unfortunately too large to be compensated with the *Doppler effect* made by separately shaking the devices. A shaking speed of $\pm$10m/s would only make a Doppler effect between $\pm$50Hz (at 2.4GHz) which probably cannot counterbalance the error ϵ.

A possible defense against this attack consists of adding a deliberate and random frequency offset so that f_A and f_B span over similar frequency ranges. This solution however requires firmware changes, making it a longer-term solution. Let the frequency offset tolerance be $\pm\delta$ and $f_A < f_B$ (i.e. f_A and f_B are within the range $[f - \delta; f + \delta]$) and both devices add a deliberate random frequency shift t to each packet. The devices will have a frequency range between $[f_A - t; f_A + t]$ and $[f_B - t; f_B + t]$ respectively. Assuming that an eavesdropper, Eve, knows f_A and f_B, then she can deduce that the packets transmitted with a frequency larger than $f_A + t$ originate from B, and the packets transmitted with a frequency smaller than $f_B - t$ originate from A.

However, the packets received within range $[f_B - t; f_A + t]$ are emitted by A or B with the same probability. This is illustrated in Figure 7. Let k be the number of packets emitted by each device. It can be shown that, $\frac{k}{2t} \times (f_A - f_B + 2t)$ packets of A (and B) will have a central frequency between $[f_B - t; f_A + t]$. If we wish to use an 80-bit secret key, at least 40 packets of A and 40 packets of B must be in this frequency range. This condition is satisfied if: $\frac{k}{2t} \times (f_A - f_B + 2t) > 40$, i.e.

$$k > \frac{40 \times 2t}{f_A - f_B + 2t}$$

where $t > \delta$ (recall that 2δ is the maximum frequency shift between two devices). In the worse scenario, $f_A = f - \delta$ and $f_B = f + \delta$, i.e. $f_A - f_B = -2 \times \delta$ and $k > 40 \times t/(t - \delta)$. If t is set to $2 \times \delta$, then k can be set to 80. To summarize, by setting t to $2 \times \delta$ and k to 80 (instead of 40 in the basic scheme), the generated secret is secure for any value of f_A and f_B in $[f - \delta; f + \delta]$.

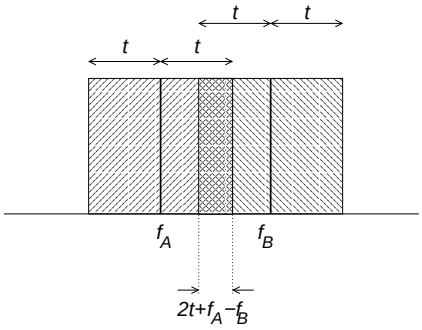


Figure 7: A technique for hiding the clock drift between two devices. In the region where the central radio frequencies of each device overlap, the packets of A and B are indistinguishable.

6.2 Camera-assisted packet captures

Another type of attack that "Shake Them Up!" may be exposed to is a video camera assisted attack. Using a signal level analyzer that is synchronized with a video camera, an attacker may correlate different device locations and their respective signal power during the shaking process. Users may employ different strategies against this threat such as hiding the sensor devices with their hand (provided that the devices are small).

Hiding the devices has also the additional benefits that it protects our scheme against eavesdroppers with unidirectional antennas. Since the location of the sensors are hidden, an eavesdropper cannot aim an antenna at one of the sensors for identifying its packets.

7 Conclusion

In this paper we presented a novel secure pairing scheme for CPU-constrained devices without a need for special hardware or interfaces. Using an existing communication channel such as 802.11 or 802.15.4, two devices that are close to each other can agree on a secret key using an algorithm that does not depend on CPU-intensive operations. On the other hand, user assistance is required for *shaking* the devices during key agreement in order to preserve key secrecy.

One alternative could consist of randomly varying the signal level (in software) during key agreement. However, this solution is not secure because an eavesdropper may aim an unidirectional antenna at one device, identify its packets and therefore retrieve the secret key. Furthermore we have discovered by experimentations that if the two devices are not shaken, one of them can mask the signal of the other one and attenuate its transmission power significantly. Consequently, the packets from each device are received at a different signal level, and the secret key can easily be retrieved. Shaking solves all these problems. It seems to be the only solution that can address all kinds of RSSI-based signal level analysis threats that our key agreement protocol may face. The proposed protocol works with off-the-shelf 802.11 wireless cards and is secure against eavesdropping attacks that use power analysis. It requires, however, some firmware changes to protect against attacks that attempt to identify the source of packets from their transmission frequency.

One limitation of our scheme is that it is specific to random media access technologies. For example, it is not suitable for TDMA-based protocols and, therefore, cannot be used with Bluetooth devices. Our scheme requires CSMA-based systems, such as 802.11 or 802.15.4 (an emerging Wireless PAN technology, designed for low power sensors). Another noticeable limitation is that it requires that the transmission power of both devices be similar. This was the case with the 802.11 devices that we used for our experimentations. However, for some wireless technologies, a power control protocol might be required to adjust the transmission power accordingly.

Objects with microprocessors and wireless transceivers surround us. Today's users are more and more technology and security-aware. Almost all users today learned that a system access password should contain non-alphanumeric characters. We have learned (or are forced to learn) how to handle computer viruses. Technology and information security have become part of our everyday lives. Thus, we believe that future users can also learn that *two small devices must be shaken well before secure use*. This is actually a very common protocol that we already execute everyday. For example, orange juice or shaving cream bottles are universally shaken/moved before usage. This is now a well-known and quite a natural "protocol". Furthermore it is commonly accepted that it is the responsibility and in the interest of the consumers to perform this shaking operation properly. Similarly, in our case by shaking the devices well, the user can make sure that the two devices are paired securely.

Acknowledgement

We would like thank Roy Want, Gene Tsudik and the anonymous reviewers for their excellent remarks that helped us improve this paper.

References

[1] Fundamentals of Quartz Oscillators. HP Applica-

tion Note 200-2.

[2] http://www.telluriantech.com. Specialty Crystals, Quartz Crystals.

[3] ALPERN, B., AND SCHNEIDER, F. Key exchange using "Keyless Cryptography". *Information processing letters 16*, 2 (February 1983), 79–82.

[4] CHAYAT, N. 802.11a PHY Overview. Slides available at: http://www.nwest.nist.gov/mtg3/papers/chayat.pdf.

[5] DAI, W. Speed benchmarks for various ciphers and hash functions. URL:http://www.eskimo.com/∼weidai/.

[6] DIFFIE, W., AND HELLMAN, M. New directions in cryptography. *IEEE Transactions on Information Theory IT-22*, 6 (1976), 644–654.

[7] GEHRMANN, C., AND NYBERG, K. Enhancements to bluetooth baseband security. In *Nordsec'01* (Kopenhagen, Denmark, November 2001).

[8] GOLDWASSER, S., AND BELLARE, M. Lectures notes in cryptography. URL:http://www.cs.ucsd.edu/users /mihir/papers/gb.html.

[9] HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D. E., AND PISTER, K. S. J. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems* (2000), pp. 93–104.

[10] HOEPMAN, J.-H. Ephemeral pairing in anonymous networks. Available at: http://www.cs.kun.nl/∼jhh/publications/anon-pairing.pdf.

[11] HOEPMAN, J.-H. The ephemeral pairing problem. In *8th Int. Conf. Financial Cryptography* (Key West, Florida, February 9-12 2004), pp. 212–226.

[12] HOLMQUIST ET AL, L. A. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In *Ubicomp 2001* (Atlanta, Georgia, September 30, October 2 2001).

[13] KARLOF, C., SASTRY, N., AND WAGNER, D. Tinysec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)* (November 2004).

[14] LENSTRA, A. K., AND VERHEUL, E. R. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research 14*, 4 (2001), 255–293.

[15] LESTER, J., HANNAFORD, B., AND G., B. "Are You with Me? - Using Accelerometers to Determine If Two Devices Are Carried by the Same Person". In *Pervasive 2004* (Vienna, Austria, April 21-23 2004).

[16] MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. *Handbook of applied cryptography*. CRC Press series on discrete mathematics and its applications. 1997. ISBN 0-8493-8523-7.

[17] OGILVIE, B. Clock Solutions for WiFi (IEEE 802.11). Saronix(tm) application note, 2003.

[18] RIVEST, R., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Coomunications of the ACM 21* (1978), 120–126.

[19] STAJANO, F., AND ANDERSON, R. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols* (1999), pp. 172–194.

[20] VIG, J., AND BALLATO, A. Frequency Control Devices. Reprinted from Ultrasonic Instruments and Devices, Academic Press, 1999.

[21] WANT, R., AND PERING, T. New Horizons for Mobile Computing. In *First IEEE International Conference on Pervasive Computing and Communication (PerCom'03)* (Dallas, Texas), pp. 3–8.

Notes

[1] Since infrared channels require line-of-sight links, they cannot be efficiently used for the actual communication between the sensors.

[2] We assume that packets do not carry information that can help identify the source address. Thus we concentrate our efforts on temporal and spatial indistinguishability problems.

[3] Available at: http://www.hpl.hp.com/personal/Jean_Tourrilhes

[4] Note that the spatial indistinguishability property requires that the two devices set the same transmission power. We observed that almost all vendors set it to 15 dBm. Otherwise, the devices should modify their transmission power to a specified value and keep it constant during key agreement.

Annexe D

Identification Privée des étiquettes RFID

Cette annexe contient un article qui a été publié dans la conférence *RFIDSec, Malaga, Espagne, Mars 2007*.

Cet article présente un nouveau protocole d'identification $ProbIP$ (Probabilistic Identification protocol). Il propose une nouvelle approche au problème de l'identification secrète des étiquettes RFID. $ProbIP$ ne nécessite très peu de calcul de la part des étiquettes et est, par conséquent, bien adaptée aux étiquettes de type EPC. Sa sécurité repose sur un problème NP-complet.

Secret Shuffling: A Novel Approach to RFID Private Identification

Claude Castelluccia and Mate Soos

INRIA, 655 avenue de l'Europe, Montbonnot, France
{claude.castelluccia, mate.soos}@inrialpes.fr

Abstract. This paper considers the problem of private identification of very small and inexpensive tags. It describes a novel scheme that does not require any computation from the tag. The proposed scheme relies on an NP-complete problem and as such is proven to be difficult to breach. We show that our solution outperforms existing computation-free schemes such as the pseudonym-rotation scheme proposal by Juels et al.[1].

1 Introduction

An RFID (Radio-Frequency Identification) tag is an extremely small electronic device that can – within a short range – wirelessly communicate with a reader. There are various types of RFID tags, ranging from very powerful to very weak devices. This paper focuses on tags with very limited computation capabilities, such as EPC tags. These devices are powered by the reader's electromagnetic field, and so need no battery and subsequently no recharging. EPC RFID tags carry interesting possibilities for the end users: they could be used to return faulty items to shops without keeping receipts, or even help intelligent washing machines that know what kind of clothes are inside them. However, with these possibilities comes a price: the possibler loss of privacy. For example, anybody possessing a reader could read any passersby's tags, which can potentially reveal even the brand of his or her socks. Similarly, tracking of people would also become possible. These possibilities scare off potential adoption as was the case with the boycott of Benetton where the garment maker was forced to take off RFID tags from their clothes.

Contributions This paper considers the problem of private identification of very small and inexpensive tags that cannot perform any cryptographic operations. Our proposal is a probabilistic identification protocol (ProbIP) that does not require any computation from the tag. Our scheme resembles Juels' pseudonym-rotation scheme as presented in [1], but increases its security significantly. The presented scheme is an identification scheme. As such, it does not address authentication, and so can not be used to authenticate a tag. It simply serves to correctly identify a tag if no active attacker is present. Privacy of the tag is preserved to some extent even if an active attacker is present.

Organization This paper is structured as follows: Section 2 presents briefly the related work. Section 3 describes our identification protocol and Section 4 provides a security analysis of our protocol.

2 Related work

Existing solutions to the RFID private identification problem can be categorized as follows: hash-lock based systems, solutions based on special tags and ultra-lightweight crypto-primitives.

Hash-lock based systems have been studied deeply, interesting papers in this category include a tree-type approach from Molnar et al. [2], an optimization of key-trees by Buttyan et al. [3], a synchronization-type approach from Ohkubo et al. [4] and a mixed approach from Lu, Han et al. [5]. Although these schemes offer relatively good security, they all suffer from the same problem: the need of a secure one-way hash function on the tag.

Some solutions use special tags, that usually have a relatively good processing power, to supervise and control all communication between the regular RFID tags and the reader. The RFID blocker tag by Juels, Rivest and Szydlo in [6] is an example of such a solution. This avenue of research has the advantage of providing very strong privacy but requires that an intelligent device be present at all times when a tag is being queried.

Ultra-lightweight crypto-primitives are an interesting avenue in RFID security research. In this category are papers such as Vajda and Buttyan's paper [7] that has been studied by Li et al. in [8], and a tiny implementation of AES by Feldhofer et al. in [9]. Also in this category, is the paper that gave us the most inspiration, written by Juels and Weis [10], that introduced HB^+, a novel lightweight authentication protocol. We believe this avenue of research has the potential to provide the best solution to the proposed problems.

3 Probabilistic Identification Protocol (ProbIP)

In this section, we introduce our Probabilistic Identification Protocol (ProbIP). In ProbIP, each tag $\mathcal{T}_j$ is configured with a unique K-bit long random secret key, k_j. The key is used as a bit-vector, with $k_j[1]$ being the first bit, $k_j[2]$ being the second, etc. The reader, $\mathcal{R}$, stores all the keys that are assigned to each of the n tags.

3.1 Protocol description

The protocol, between tag $\mathcal{T}_j$, and the reader $\mathcal{R}$, is as follows:

1. $\mathcal{R}$ initiates an identification by broadcasting a `HELLO` message.
2. Upon reception of a `HELLO` message, $\mathcal{T}_j$ replies with P *packets* and a `FINISHED` message, where P is a system parameter that will be defined in the following section. A *packet* is a list of $2L$ values, $a_1, b_1, a_2, b_2 \ldots, a_L, b_L$, where a_i is a

random index from the key $a_i \xleftarrow{r} [1, K]$ that is never repeated in the same packet, and b_i is a random bit $b_i \xleftarrow{r} \{0, 1\}$ that satisfy the following equation:

$$\sum_{i=1}^{L} k_j[a_i] \oplus b_i = L/2 \tag{1}$$

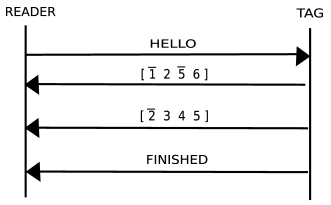
Since addition is commutative, as long as the pairs a_i, b_i for all i are not changed, the order of the pairs can change. We will note these pairs in the following fashion: $\bar{a}_i$ if $b_i = 1$ and a_i if $b_i = 0$.

3. Upon reception, $\mathcal{R}$ computes the result of eq. (1) for each packet for every tag's key in a relatively fast fashion. The key(s) that fits all the packets is suspected to have been used to send the packets.

3.2 An example

Let's consider, to illustrate our protocol, a system that uses the following artificially small system parameters, $L = 4$, $K = 6$ and $n = 4$. In this example, $\mathcal{T}_1$ is configured with the key k_1 =`011001`, $\mathcal{T}_2$ with the key k_2 =`100101`, $\mathcal{T}_3$ with the key k_3 =`011110` and finally $\mathcal{T}_4$ with k_4 =`001110`.

Let's assume that the tag the reader is trying to identify is $\mathcal{T}_2$. An example protocol run between $\mathcal{R}$ and $\mathcal{T}_2$ is the following:



In a step-by-step fashion, the following happens during this protocol run:

1. $\mathcal{R}$ broadcasts a `HELLO` message.
2. Tag $\mathcal{T}_2$ sends two packets and the `FINISHED` message. The first packet is defined by `[1̄ 2 5̄ 6]`, for which the eq. (1) wrt. k_2 is $(1 \oplus 1) + (0 \oplus 0) + (0 \oplus 1) + (1 \oplus 0) = 2 = L/2$. The second packet is defined by `[2̄ 3 4 5]` for which the eq. (1) wrt. k_2 is $(0 \oplus 1) + (0 \oplus 0) + (1 \oplus 0) + (0 \oplus 0) = 2 = L/2$.
3. Upon reception of the first packet, the reader computes for each of the 4 tags the eq. (1). $\mathcal{R}$ gets that for $\mathcal{T}_1$ it is 4, for $\mathcal{T}_2$ it is 2, for $\mathcal{T}_3$ it is 2 and for $\mathcal{T}_4$ it is 1. The reader, therefore, keeps only tags $\mathcal{T}_2$ and $\mathcal{T}_3$ as possible candidates.
4. Upon reception of the second packet, the reader computes for tags $\mathcal{T}_2$ and $\mathcal{T}_3$ the eq. (1). $\mathcal{R}$ gets that for $\mathcal{T}_2$ it is 2 and for $\mathcal{T}_3$ it is 3. At this point, tag $\mathcal{T}_2$ has been successfully identified by $\mathcal{R}$.

3.3 Minimum number of packets needed by the reader

Here, we compute the minimum amount of packets needed by $\mathcal{R}$ to correctly identify a tag. Since the protocol is probabilistic, there will always be a non-zero probability fp that the number of packets sent will not be enough. However, this probability can be arbitrary adjusted between $0 < fp < 1$.

The total number of packets possible for *all* keys is $\binom{2K}{L}$, as a_i comes from a set of size K and b_i comes from a set of size 2, whereas for a *given* key, the number of possible packets is only $\binom{K}{L/2}\binom{K-L/2}{L/2}$ since eq. (1) must hold and indices cannot be repeated in a packet. The ratio of these two numbers

$$R = \frac{\binom{K}{L/2}\binom{K-L/2}{L/2}}{\binom{2K}{L}} \tag{2}$$

is the probability that a random packet is valid for a random tag. As an example, for $K = 400, L = 10$, $R \approx 0.232$.

Given n tags, the false positive probability, fp, that p packets generated by a given tag match another tag's key can be calculated as $fp = n * R^p$. The number of packets sent from the tag to the reader should then be

$$P = \left\lceil \frac{\log(1/n * fp)}{\log(R)} \right\rceil \tag{3}$$

which is, for the example parameters of $L = 10$, $fp = 0.1$ and $n = 10^7$, $P = \lceil 12.62 \rceil = 13$. If these packets do not suffice (which has a low chance of happening), repeated identification attempts will be carried out by the reader until it finds the correct tag.

3.4 Parameters

The parameter K must be at least $\lceil \log_2(n) \rceil$ bits, but as the security of the system will rely on the condition that $n \ll 2^K$, the larger this parameter is, the more secure the system. Also, K should be at least an order of magnitude larger than L. The parameter L must be such that $L/2$ is a whole number. When deciding the parameters, the number of bits sent in one identification

$$B = P * L * (\lceil \log_2(K) \rceil + 1) \tag{4}$$

which is also the minimum amount of random bits that need to be generated during an identification, must be kept in mind. The parameters L, K and n all influence this number. As an example, for $K = 400, L = 10$ and $n = 10^7$, $B = 1300$ bits. It is important to note that sending this information is just a fraction of a second given a 52.969 kb/s label-to-interrogator link in Class 1 EPC tags [11].

3.5 Algorithm used by the reader

This section describes the algorithm used by the reader or a set of back-end servers, to identify the tag using the packets it sent. It is assumed that $\mathcal{R}$ knows the keys $k_1 \dots k_n$ of all the tags in the system. These keys are stored not in their natural order $k_j[1], k_j[2], \dots k_j[K]$ for all $\mathcal{T}_j \in \{\mathcal{T}_1, \dots, \mathcal{T}_n\}$, but in their column-like order $k_1[i], k_2[i], \dots k_n[i]$ for all $i \in [1, K]$. These K columns we will call $Col_1, \dots, Col_K$. Naturally, storing these columns needs exactly the same amount of memory as simply storing the keys, i.e. $n * K$ bits.

At each protocol instance, the following is executed by $\mathcal{R}$:

1. $\mathcal{R}$ fills with zeros a temporary n-long byte-vector *temp*. This will store the result of the eq. (1), for each tag. A byte is sufficient for any $L < 1.5 * 256$.
2. Upon reception of a packet, $\mathcal{R}$ performs the following algorithm for each of the packets' L pairs (a_i, b_i): For each tag $\mathcal{T}_j$ in the system, $temp[j]$ is incremented by one if $Col_{a_i}[j] \oplus b_i = 1$. Iteration through the *temp* and the Col_{a_i} can be parallel, so for a given index, on average $n + 8n$ bits of memory need to be read and $8n/2$ bits of memory written.
3. Once all pairs in the packet have been considered, all tags $\mathcal{T}_j$ for which $temp[j] = L/2$ could have sent the packet.
4. Steps 2-3 are repeated for all packets with different *temp*s, i.e. $temp_1$ for packet no. 1, $temp_2$ for packet no 2, etc.
5. The identified tag is the tag $\mathcal{T}_j$ for which $temp_i[j] = L/2$ for all $i \in [1, P]$.

The proposed algorithm identifies a tag by looking through, on average, $P * L * (n + 8n)$ bits of memory and, by writing $P * L * 8n/2$ bits of memory space, while doing $L * n$ comparisons and $L * n/2$ incrementation per packet, plus $P * n$ comparisons for evaluating all the packets' results in step 5, which gives $P * L * (n + n/2) + P * n$ processing steps in total. For instance, if $L = 10, K = 400, n = 10^7$ then the overall memory requirement is $400 * 10^7 + 13 * 8 * 10^7$ bits= 630MB and the overall processing requirement is $13 * 10 * (1.5 * 10^7) + 13 * 10^7 \approx 2.08e9$ processing steps. Parallelization of this algorithm is relatively simple, and can bring down both the memory and processing requirement of individual computers.

Note that an adversary does not know the configured set of keys, and would need to run this algorithm with $n = 2^K$ and so $\approx 1.63e113$ GB of memory would be needed for the same parameters ($L = 10, K = 400$). The processing need would increase to similar proportions.

4 Security analysis of ProbIP

In this section we will evaluate the security of our scheme using the "strong privacy" model proposed by Juels and Weis in [12]. In our scheme, *tags' keys are completely independent of each other* thus the corruption of one tag does not affect the security of the rest of the system. Therefore, it is useless for adversary

$\mathcal{A}$ to use the the SetKey procedure to change the key of tags. It is also useless for $\mathcal{A}$ to examine any other tags than the ones it will pick, i.e. $\mathcal{T}_A$ and $\mathcal{T}_B$. In our scheme, ReaderInit is a simple fixed `HELLO` message, so it need not be executed by $\mathcal{A}$ at all. Therefore, in view of our protocol, the privacy experiment of the model can be refined to what is present in Fig. 1.

Experiment $\mathbf{Exp}^{priv}_{\mathcal{A},\mathcal{S}}[K, n, x_A + x_B + x_C]$:

Setup:

(1) Generate keys $(k_1, \ldots, k_n)$ uniquely and randomly with GenKey
(2) Initialize $\mathcal{R}$ with keys $(k_1, \ldots, k_n)$
(3) Set each $\mathcal{T}_i$'s key k_i with a SetKey call

Phase 1 (Learning):

(4) Let $\mathcal{A}$ perform x_A TagInit calls with $\mathcal{T}_A$ and let it record the received packets into the set X_A
(5) Let $\mathcal{A}$ perform x_B TagInit calls with $\mathcal{T}_B$ and let it record the received packets into the set X_B

Phase 2 (Challenge):

(6) Let $\mathcal{T}_C \xleftarrow{r} \{\mathcal{T}_A, \mathcal{T}_B\}$
(7) Let $\mathcal{A}$ perform x_C TagInit calls with $\mathcal{T}_C$ and let it record the received packets into the set X_C
(8) Let $\mathcal{A}$ perform calculations on the recorded packets in order to make an educated guess whether $\mathcal{T}_C = \mathcal{T}_A$ or $\mathcal{T}_C = \mathcal{T}_B$.

Exp succeeds if $\mathcal{A}$ guessed $\mathcal{T}_C$ correctly

Fig. 1. The privacy experiment as proposed by Juels and Weis in [12], refined to the specifics of our protocol

In order to find out how the experiment can be optimized by the adversary and how he should choose the parameters x_A, x_B and x_C, we will first analyze what a packet is. Then we propose an algorithm to perform the attack, and finally, we will experimentally show what is the resistance of our proposed protocol for certain parameter combinations.

4.1 A closer look at the packets

Looking at the packets in a more mathematical way, they describe an $L/2$-in-L LSAT problem. In $L/2$-in-L LSAT, like in LSAT, the input instance is a collection of clauses, where each clause consists of exactly L literals, where each literal is either a variable or its negation. The $L/2$-in-L LSAT problem is

to determine whether there exists a truth assignment to the variables so that each clause has exactly $L/2$ true literals. This problem is NP-complete if $L > 2$ as indicated by Schaefer's dichotomy theorem [13]. Thus, if an attacker, given enough packets, wants to solve for k, if $L > 2$, he would have to solve an NP-complete problem.

Similarly, a packet can also be looked at as an L-long Linear Pseudo-Boolean Constraint (LPBC) as the original eq. (1) implied.

4.2 Algorithm used by the attacker

In light of what the true nature of a packet is, the way for an attacker to attack our scheme in the given model is to execute an LPBC solver on $X_C \cup X_A$, and examine the output of the solver. If the result is unsatisfiable (UNSAT), then surely $\mathcal{T}_C \neq \mathcal{T}_A$ (consequently, $\mathcal{T}_C = \mathcal{T}_B$), since $\mathcal{T}_A$ only sends packets that have a solution, k_A. However, if the result is satisfiable (SAT), $\mathcal{T}_C$ can be either of the tags. But, if the attacker knows that he has gathered enough packets that had $\mathcal{T}_C \neq \mathcal{T}_A$ the result would surely have been UNSAT, then he knows that $\mathcal{T}_C = \mathcal{T}_A$. If he cannot gather this amount of packets in X_A and X_C, then he will almost always get the result SAT, which he cannot use. Therefore there is no need for him to gather packets from more than one of the tags in the learning phase, and it does not matter which one he picks. We decided to pick $\mathcal{T}_A$.

While keeping the above argument in mind, one could reason in the following way about NP-hard problems:

1. They are hard to solve exactly but can in certain cases be approximated easily. While this is true, it does not help the attacker in any way. If the attacker can find an approximate solution to the set of constraints defined by $X_A \cup X_B$, that does not tell anything about whether $\mathcal{T}_C = \mathcal{T}_A$ or $\mathcal{T}_C \neq \mathcal{T}_A$.
2. They are hard to solve in the worst case, but easy to solve most of their instances. This is true, and we will deal with this later, where we show that the problem generated by the tag is exactly what is suggested in [14] and which is known to be hard.

LPBC solver used We tried two of the most respected LPBC solvers, Minisat [15] and Toolbar [16], and a less known one, Galena [17]. Each one had its own distinct advantage. Galena was designed to solve exactly these kinds of problems, Toolbar had a large user-base and thus good support, while Minisat was the best performing in many SAT competitions. Minisat had multiple advantages over the competition: since it was a pure SAT solver, it could use the well-established DPLL algorithm as described by Davis, Putnam et al. in [18], and could also benefit from "learning" as first shown by Schulz et al. in [19], and improved by Marques Silva et al. in [20]. Galena adapted these techniques to solving LPBC problems, but it could not use the well-established literature of how to implement them in a way that is fast.

The fastest solver by far (and also, the most modifiable), was Minisat. It was the winner of multiple SAT-competitions in 2005 and came the first in the 2006 SAT-race, and so gave a good foundation to base our results on. One would expect that the speed slowdown of using the LPB constraints of the type in eq. (1) converted into regular CNF formulas is significant. However, as was shown by the version of Minisat called "Minisat+" this method is faster than directly tackling the original LPB-constraint problem: Minisat was much better than its rivals in the LPB-part of the SAT competition in 2006.

In order to use Minisat, we needed to convert each packet to a set of CNF clauses. We thus used a conversion that made from e.g the packet of `[1 2̄ 5 9]` the following CNF clauses:

$$(1 \vee \bar{2} \vee 5) \wedge (\bar{1} \vee 2 \vee \bar{5}) \wedge (\bar{2} \vee 5 \vee 9) \wedge (2 \vee \bar{5} \vee \bar{9}) \wedge \ldots$$

i.e. all $L/2 + 1$ combinations in their original and their negated form. These formulas simply mean that any $L/2 + 1$ literals that have the same value is forbidden. We used this conversion because the conversion algorithm included in Minisat+ converted the problem in such a way that it was slower than using the conversion given above.

Minimum number of packets needed by the attacker Let's assume that $\mathcal{T}_A \neq \mathcal{T}_C$. The attacker first needs to know how should he distribute the number of ReaderInit calls he has between $\mathcal{T}_A$ and $\mathcal{T}_C$ in order to have the highest possibility of finding out that $\mathcal{T}_A \neq \mathcal{T}_C$. We will now calculate this ratio and deduce equation for the minimum amount of packets needed.

x_A packets from $\mathcal{T}_A$ reduces the number of possible keys by a factor of R^{x_A}, the remaining set of possible keys we call S_A . x_C packets from $\mathcal{T}_C$ also reduces the number of possible keys by a factor of R^{x_C}, the remaining set of possible keys we call S_C . Intuitively, the union of these packets will leave the attacker with

$$PC = |S_A \cap S_C| = \max \begin{cases} \max(2^K * R^{x_A}, 1) * (2^K * R^{x_C}) \\ \max(2^K * R^{x_C}, 1) * (2^K * R^{x_A}) \end{cases} \quad (5)$$

number of possible key combinations. This equation tells two things. Firstly, the intersection is the smallest if $x_A = x_C$. Secondly, in case $\mathcal{T}_A \neq \mathcal{T}_C$, the closer PC is to 0, the higher the possibility of the attacker to find that the constraints $X_A \cup X_C$ is UNSAT. As an example, if $PC = 0.1$ then he has a 90% chance that he will find out that the resulting constraints are UNSAT (thus $\mathcal{T}_A \neq \mathcal{T}_C$) and if he finds that $X_A \cup X_C$ is SAT, then he can be 90% sure that indeed $\mathcal{T}_A = \mathcal{T}_C$.

Therefore, if the attacker wishes to attain a 90% chance ratio of finding out which tag is $\mathcal{T}_C$ then he needs

$$P_{att} = \left\lceil \frac{\log(1/2^K * 0.1)}{\log(R)} \right\rceil \quad (6)$$

packets equally distributed between x_A and x_B. As an example, if $K = 400$ and $L = 10$, then $P_{att} = \lceil 191.62 \rceil = 192$. If $n = 10^7$ and consequently, $P = 13$, this

is $\lceil P_{att}/P \rceil = 15$ identifications in total, i.e. 8 identifications for both $\mathcal{T}_A$ and $\mathcal{T}_C$. It is interesting to observe that the equations (6) and (3) completely match if $fp = 0.1$ and $n = 2^K$ i.e. the search space of the reader is not restricted to the configured set of keys.

The threshold phenomenon As it is hypothesized by Cheeseman et al. in [14] and further explained by B. Smith in [21], all NP-hard problems exhibit a so-called phase-transition, which states that given a randomly generated NP-hard CSP(Constraint Satisfaction Problem), there is always a point where it is the hardest to solve the generated problem, and this corresponds exactly to the point where there is a transition from SAT to UNSAT. From this point on, the difficulty of finding a solution decreases at an exponential rate, along with the possibility of having any solution at all.

This phenomenon plays a crucial part in the security evaluation of our protocol: as the possibility of UNSAT increases if $\mathcal{T}_A \neq \mathcal{T}_C$, the hardness of finding this out increases as well. The peak of the difficulty of finding out UNSAT is then at the minimum amount of packets the attacker needs (i.e near P_{att}) to find that the CPS defined by the packets is indeed UNSAT. If more packets are known by the attacker, this difficulty decreases at an exponential rate in relation to the extra number of packets gathered.

Precisely calculating the phase-transition and its corresponding graph is very hard for a given CSP: even for such a widely studied problem as k-SAT, since its inception in 1991 with the influential paper of Cheeseman et al. [14] it has taken more than 10 years to prove that the threshold for k-SAT is $2^k \log 2 - O(k)$ [22]. So, instead of mathematically calculating the threshold and its corresponding graph, we will experimentally show it using Minisat, and extrapolating the results, deduce the protocol's resistance to attacks.

An example threshold plot is present in Fig. 2. In this particular instance of the protocol the parameters were such that $P_{att} \approx 36$ and $\mathcal{T}_A = \mathcal{T}_C$ so that the Hamming distance of solution given by the satisfiability solver approached 0 as the number of packets given to it approached P_{att}. The phase-transition is clearly between no. of packets 30 and 36.

4.3 Results

In calculating the resistance to attacks, we used the parameter L =10. Knowing that it is hardest to solve at P_{att}, we gave our simulated attacker multiple number of P_{att} packets to evaluate the computation speed-up it can achieve if given more packets. All experiments were performed with a 3GHz Pentium-D machine with 2 GB of memory. The plot for 64, 192, 576 times P_{att} is in Fig. 3.

The scale on the time axis is logarithmic, since the time to break the system increases exponentially as the key-length is increased. This is a consequence of using an NP-hard problem to base the security of the protocol on. The breaking times for different values of P_{att}, n and K are shown in Table 1.

The results clearly show that there is a trade-off between the number of packets collected and the hardness of breaking the system privacy (i.e. winning

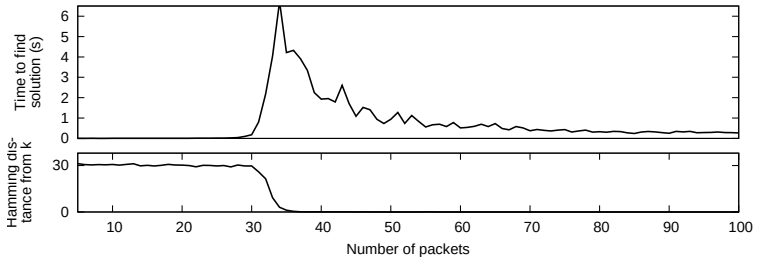


Fig. 2. The threshold phenomenon illustrated. Top part of the plot shows the time to find a solution versus information given, the bottom part shows the Hamming distance of the solution found from the key of the tag

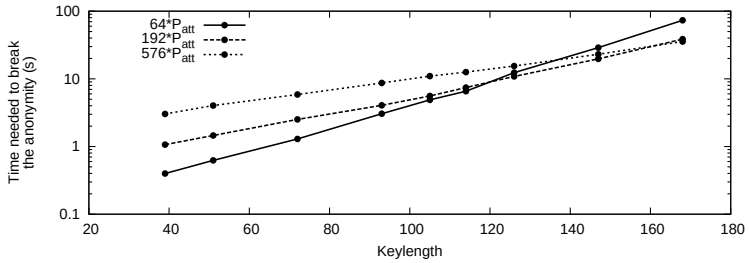


Fig. 3. Time to break the anonymity versus the keylength for different $overinfo$ rates (i.e. the number of times P_{att} packets was given to the attacker)

the privacy experiment). The more packets an attacker can collect, the easier it is for him to break the system.

4.4 Analysis

The results clearly show that there is a trade-off between the number of packets collected and the hardness of breaking the system privacy (i.e. winning the privacy experiment). The more packets an attacker can collect, the easier it is for him to break the system. This interesting property is a direct result of the threshold phenomenon.

An interesting property of our scheme is that n is not directly present anywhere in the results. This is because the only help a greater n gives to the attacker is that the number of packets per identification, P, will be greater, and so P_{att} number of packets will be attainable by less TagInit queries.

packets/K	100	200	400	1000
$1*P_{att}$	1.47e2 s	3.17e11 s	1.46e28 s	1.46e78 s
$3*P_{att}$	3.33e1 s	7.41e5 s	3.67e14 s	4.49e40 s
$9*P_{att}$	6.31e0 s	4.54e3 s	2.35e9 s	3.27e26 s
$27*P_{att}$	4.27e0 s	6.37e2 s	1.42e7 s	1.57e20 s
$64*P_{att}$	4.02e0 s	4.87e2 s	7.15e6 s	2.27e19 s
$192*P_{att}$	5.34e0 s	7.31e1 s	1.37e4 s	9.01e10 s
$576*P_{att}$	1.00e1 s	7.28e1 s	3.86e3 s	5.74e8 s

Table 1. This table shows the trade-off in time and calculations that are possible in terms of the $overinfo$ rate (i.e. the number of times P_{att} packets was given to the attacker). Larger keysizes make the attacker's job more difficult - he will either need to do much more calculations or gather much more packets than with smaller keysizes.

Our scheme, unlike the usual cryptographic schemes, is simple to analyze. The detailed examination of the problem that leads to the results shows that any algorithm that can break the privacy of the proposed protocol significantly faster than the presented one is a technological breakthrough.

Comparison with other schemes The presented scheme is targeted for very simple RFID tags with no computational capabilities. Therefore, it is not meant to compete with schemes that use cryptographic functions such as a secure hash. As such, our scheme should be compared to schemes such as the pseudonym-rotation scheme, proposed in [1], where each tag is configured with a short list of random identifiers called pseudonyms. In this scheme, each time a tag is queried, it emits the next pseudonym in the list, cycling to the beginning when the list is exhausted. With this scheme, if $n = 10^7$ a tag that contains 15 pseudonyms which corresponds to a memory size of $15 * \lceil \log_2(10^7 * 15) \rceil = 420$ bits, loses its privacy after 16 requests by the attacker. Note that this is exactly the same as our protocol with parameters $K = 420, L = 10, \lceil P_{att}/P \rceil = 16$, with the exception that in our protocol, the attacker needs not only to *record* the sent information, but also to execute a difficult *computation* to win the privacy experiment. In other words, our protocol gives the same amount of *information* during one protocol run as the pseudonym-rotation scheme, but in a coded way. As such, for an attacker to break our scheme in a reasonable amount of time, he needs to collect much more information than would be necessary from a purely information theoretic point of view.

References

1. Juels, A.: Minimalist cryptography for low-cost RFID tags. In: SCN 2004
2. Molnar, D., Soppera, A., Wagner, D.: A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In: SAC 2005
3. Buttyán, L., Holczer, T., Vajda, I.: Optimal key-trees for tree-based private authentication. In: PET 2006

4. Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient hash-chain based RFID privacy protection scheme. In: Ubicomp 2004, Workshop Privacy: Current Status and Future Directions
5. Lu, L., Liu, Y., Hu, L., Han, J., Ni, L.: A dynamic key-updating private authentication protocol for RFID systems. In: PerCom 2007
6. Juels, A., Rivest, R., Szydlo, M.: The blocker tag: Selective blocking of RFID tags for consumer privacy. In: ACM CCS 2003
7. Vajda, I., Buttyán, L.: Lightweight authentication protocols for low-cost RFID tags. In: Ubicomp 2003
8. Li, T., Wang, G.: Security analysis of two ultra-lightweight RFID authentication protocols. In: IFIP SEC 2007
9. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: Aes implementation on a grain of sand. In: Information Security, IEEE (2005) 13–20
10. Juels, A., Weis, S.: Authenticating pervasive devices with human protocols. In: CRYPTO'05
11. EPCglobal: 13.56 mhz ism band class 1 radio frequency identification tag interface specification (2003). Technical report, Auto-ID cetner, MIT
12. Juels, A., Weis, S.: Defining strong privacy for RFID. Cryptology ePrint Archive, Report 2006/137 (2006)
13. Schaefer, T.J.: The complexity of satisfiability problems. In: STOC '78
14. Cheeseman, P., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In: IJCAI-91
15. Een, N., Soorensson, N.: An extensible sat-solver [ver 1.2]. Theory and Applications of Satisfiability Testing **2919/2004**
16. Bouveret, S., Heras, F., de Givry, S., Larrosa, J., Sanchez, M., Schiex, T.: Toolbar: a state-of-the-art platform for wcsp (2004)
17. Chai, D., Kuehlmann, A.: A fast pseudo-boolean constraint solver. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2003. 305–317
18. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**(3) (1960) 201–215
19. Schulz, M.H., Auth, E.: Improved deterministic test pattern generation with applications to redundancy identification. IEEE Transactions on computer-aided design **8**(7) (July 1989) 811–816
20. Silva, J.P.M., Sakallah, K.A.: Graspa new search algorithm for satisfiability. In: ICCAD '96
21. Smith, B.: The phase transition in constraint satisfaction problems: A CLoser look at the mushy region. In: Proceedings ECAI'94. (1994)
22. Achlioptas, D., Peres, Y.: The threshold for random k-sat is 2ˆ k(ln 2 - o(k)). In: STOC '03

Annexe E

Les Identifiants Cryptographiques

Cette annexe contient un article qui a été publié dans la revue *ACM Transactions on Information and System Security (TISSEC), Volume 7, Numéro 1, pages 97-127 en Février 2004*. Il présente les concepts des addresses et identifiants cryptographiques. Ces adresses et identifiants sont cryptographiquement vérifiables. Cette propriété permet de résoudre plusieurs problèmes de sécurité, allant du vol d'adresse IP à la sécurité des groupes.

Crypto-Based Identifiers (CBIDs): Concepts and Applications

GABRIEL MONTENEGRO
Sun Labs, Europe, France
and
CLAUDE CASTELLUCCIA
INRIA Rhône-Alpes, France

This paper addresses the identifier ownership problem. It does so by using characteristics of Statistical Uniqueness and Cryptographic Verifiability (SUCV) of certain entities which this document calls SUCV Identifiers and Addresses, or, alternatively, Crypto-based Identifiers. Their characteristics allow them to severely limit certain classes of denial-of-service attacks and hijacking attacks. SUCV addresses are particularly applicable to solve the *address ownership* problem that hinders mechanisms like Binding Updates in Mobile IPv6.

Categories and Subject Descriptors: C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks—*General security and protection (e.g., firewalls)*

General Terms: Security, Verification

Additional Key Words and Phrases: Security, mobile IPv6, address ownership, group management, authorization, opportunistic encryption

1. INTRODUCTION

This paper addresses the identifier ownership problem [Nikander 2001] by using characteristics of Statistical Uniqueness and Cryptographic Verifiability (SUCV) of certain entities which this document calls SUCV Identifiers (SUCV IDs) or, alternatively, Crypto-based Identifiers (CBIDs). This paper also proposes using these SUCV characteristics in related entities called SUCV addresses in order to severely limit certain classes of denial-of-service (DOS) attacks and hijacking attacks. SUCV addresses can solve the *address ownership* problem that hinders mechanisms like Binding Updates (BUs) in Mobile IPv6. We first examine this problem and our proposed solution in some depth.

A preliminary version of portions of this material appeared in G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," Proceedings of the 2002 Network and Distributed System Security Conference (NDSS02), San Diego, February 2002.
Authors' addresses: G. Montenegro, Sun Labs, Europe, 180 Avenue de l'Europe, ZIRST de Montbonnot, 38344 Saint Ismier cedex, France; email: gab@sun.com; C. Castelluccia, INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 Montbonnot, France; email: claude.castelluccia@inrialpes.fr.

Subsequently, we explain other applications of the same concept. In particular, we explain a powerful construct obtained by adding the expressiveness of authorization certificates to unforgeable CBID identities.

This paper is structured as follows. Section 2 defines the *address ownership* problem. Section 3 presents the notation used throughout this paper. Section 4 gives an overview of our proposal. Section 5 presents SUCV identifiers and addresses, and how to generate them. Section 6 describes SUCV Protocol (sucvP), the protocol that is used by a mobile node to prove ownership of its addresses to its correspondent nodes and to generate session keys. Section 7 presents an extension to sucvP for constrained devices (PDAs, sensors, phones, etc.). Section 8 explains how to provide data confidentiality and location privacy with sucvP. Section 9 presents a security analysis of our proposal. Section 10 presents other applications of the SUCV properties of CBIDs and of their use as an authorization mechanism. Section 11 presents a human-friendly protocol to bootstrapping CBIDs as an alternative to existing mechanisms. Section 12 compares our scheme with related work. Finally Section 13 concludes.

2. PROBLEM STATEMENT: THE NEED FOR PROOF OF ADDRESS OWNERSHIP

Nikander [2001] argues that there is a fundamental problem in handling operations like BUs in Mobile IP for IPv6 [Johnson et al. 2003], source routing, and so on, that allows hosts to modify how other hosts route packets to a certain destination. The problem is that these operations can be misused by rogue nodes to redirect traffic away from its legitimate destination. Authentication does not solve this problem. Even if a node unequivocally identifies itself, this has no bearing on its rights to modify how packets to any given address are routed. This is true even if its packets currently seem to emanate from the address in question. This last point is obvious if one considers DHCP leased addresses. It is imperative not to allow any node to redirect traffic for a DHCP address for which it held a valid lease previously. This would allow it to hijack traffic meant for the current valid user of the address in question. Hence, protection against hijacking of valid addresses requires cryptographic authorization for operations that modify routing (BUs, source routing, etc.). One way to achieve authorization is by showing that the requesting node owns the address for which routing information is being altered. Quoting from Nikander [2001]: "Currently there exists no specified mechanism for proving address ownership in Internet-wide scale."

3. NOTATION

This section presents the notation used throughout this paper.

—*prf*: Pseudo-random function. SUCV mandates the use of the HMAC-SHA-1 construct of the keyed hash function HMAC [Krawczyk et al. 1997], which produces 160 bits of output. Input key is assumed to also be 160 bits.

—*prfT*: Pseudo-random function whose output is truncated by taking the T leftmost bits of the output. In SUCV, HMAC-SHA1 is used, so prf96, for

example, would be the keyed hash function HMAC-SHA1-96 [Madson and Glenn 1998].

—*hash*: Cryptographic hash function, SHA-1 [NIST 1995] for SUCV.

—*hashT*: Cryptographic hash function whose output is truncated by taking the T leftmost bits of the output.

—*SUCV*: Statistical uniqueness and cryptographic verifiability, the property exhibited by the identifiers and addresses which are the subject of this study. We also use SUCV to refer to the resultant mechanism as a whole.

—*sucvP*: The protocol developed here, whose objectives are proof of address ownership and session key generation.

—*sucvID*: 128-bit identifier obtained as the keyed hash output of the hash of the public key, using an imprint value as the input key.

—*sucvHID*: 64-bit SUCV identifier used instead of the *interface identifier*, and combined with the routing prefix to form an autoconfigured IPv6 address [Hinden and Deering 2003]. Obtained as the keyed hash output of the hash of the public key, using an imprint value as the input key.

—*CBID*: Crypto-based ID. This general term denotes any entity that has the SUCV property: either a pure ID (SUCV ID) or an address (SUCV address).

—*MIPv6*: Mobile IPv6 [Johnson et al. 2003].

—*MN*, *HA*, *CN*, *BU*, *BA*, and *CoA*: Abbreviations of mobile node, home agent, correspondent node, binding update, binding acknowledgement, and care-of address, respectively, as defined by MIPv6 [Johnson et al. 2003].

4. PROPOSAL OVERVIEW

We assume that we have a network in which the nodes inherently distrust each other, and in which a global or centralized Public Key Infrastructure (PKI) or Key Distribution Center (KDC) is not available. The goal is to arrive at some fundamental assumptions about trust on top of which one can build some useful peer-to-peer communication using opportunistic security. But in such a network, is there a default rule we can follow safely? We posit this is it:

Default Trust Rule. Redirect operations are allowed only with addresses that are securely bound to the requesting entity.

The above rule constitutes the only rule that operates by default, allowing any other more dangerous operation only if authorized by strong cryptographic mechanisms. In the absence of a trusted third party, how does a principal prove ownership of its identity to a peer? Notice that usual owner verification relies on a third party to provide this function. In our proposal, the principal self-generates a private/public key pair. However, it is much more practical for protocols to use fixed length identifiers (representations of identities). Because of this, we do not use the public key itself as the identifier. Instead, the principal uses material obtained via a *prf* of the public key as its identity (or as part of its address), and proves its ownership by signing it with its private key. The recipient verifies the signature, and, consequently, the ownership of the

identity. These considerations lead to the following fundamental assumption with respect to the above Default Trust Rule:

Refined Default Trust Rule. It is safe to allow entities that have the SUCV property to affect routing via redirect operations.

5. SUCV IDENTIFIERS AND ADDRESSES

In MIPv6, a node starts using its home address, and issues BUs as it moves. Handling these BUs securely is the issue. It is never evident to the CN that whoever was using an address actually owns it. At the very most, the MN can prove that at some point it was using a certain address, but it cannot prove ownership. Ignoring this subtle distinction leads to DOS and hijacking attacks.

5.1 SUCV Identifiers

The idea is to use identifiers that have a strong cryptographic binding with their public components (of their private–public keys). This is exactly the purpose that certificates have. Let us call them SUCV IDs. Because of this, once a CN obtains information about one of these identifiers, it has a strong cryptographic assurance about which entity created it. Not only that, it knows that this identifier is owned and used exclusively by one node: its peer in the current exchange.

Using identifiers that satisfy the SUCV conditions outlined above, it is possible to gain the tremendous advantage that other nodes can safely believe the node when it claims ownership of that identifier. Hence they can safely heed its redirects when it says that it is now available at some different CoA (and later at another).

What should one use: pure identifiers with no routing significance or addresses? With pure identifiers, routing information must be included somewhere else in the packet. This takes up extra space in the packet via home address options, routing headers, or tunneling headers.

A major advantage of using an address is that the data traffic need not carry extra information in the packet to guarantee proper delivery by routing. Because of this it is useful to create addresses that are both routable and satisfy the SUCV property: *SUCV addresses*.

5.2 SUCV Addresses

In IPv6, addresses that satisfy the SUCV property may be obtained as follows (as it turns out, this is very similar to, and was predated by O'Shea and Roe [2001]):

—use the top 64 bits from your routing prefix (as in Narten and Draves [2001])
—define the bottom 64 bits as an SUCV ID (called the sucvHID). Use these 64 bits instead of the *interface identifier* in IPv6 [Hinden and Deering 2003].

The resultant 128-bit field is an identifier that is also routable, avoiding the need to take extra space in the packet by sending routing options. Notice that even after moving, it is possible to reuse the sucvHID portion of the address

with the new network prefix at the new location. Thus it is possible to reuse the HID with different CoAs.

Nevertheless, by snooping on BUs, it is possible for an attacker to learn the original network prefix used by the home address. This tells an eavesdropper where this home address began to be used, and to which network it belongs, potentially important information.

On the other hand, if you use a *pure* SUCV ID (without any routing significance), then your packets will always need extra information somewhere else to assure they are routed properly. Eavesdroppers may still know where that identity is at any particular point in time. Nevertheless, from the point of view of privacy this is a tangible improvement over always including a valid 64-bit prefix, as this divulges information about an identity's topological connectivity or under what prefix a given identity began to be used (see Section 8).

5.3 Generating SUCV Identifiers and Addresses

Identifiers and addresses for use with SUCV are generated as follows:

$$\text{sucvID} = \text{hmac-sha-1-128}(\text{sha1}(\text{imprint}), \text{sha1}(\text{PK})),$$
$$\text{sucvHID} = \text{hmac-sha-1-64}(\text{sha1}(\text{imprint}), \text{sha1}(\text{PK})),$$

where:

—*imprint*: The imprint is a 64-bit field. It could be a quantity that depends on the MNs location or something created by the MN itself (e.g., a random value). The objective is to use the imprint to limit certain types of brute-force attacks (see Section 9.1 by limiting their applicability, or by forcing interaction with the MN).

—*PK*: The public key (possible types of public keys include DSA, RSA, Elliptic Curves, etc.).

Note that according to Hinden and Deering [2003], the leftmost three bits of the sucvID can be used to unequivocally distinguish them from IPv6 addresses. Accordingly, we assume only 125 bits may be used. Additionally, bit 6 of the sucvHID (the universal/local bit) has to be set to zero to indicate that the sucvHID is not guaranteed to be globally unique. Bit 7 (the individual/group bit) must also be set to zero.

6. SUCV PROTOCOL (SUCVP) OVERVIEW

The following illustrative protocol, sucvP, is meant as an example of how the SUCV property can be used to secure what amount to *routing redirects* in IPv6, that is, Mobile IPv6 route optimization BUs. Accordingly, sucvP is run between an MN and an arbitrary CN.

It is used

—by the MN to prove ownership of its home address and optionally of its CoA,

—to establish an IPsec ESP security association (Skey, Lifetime, SPI) between the MN and the CN that will be used to secure MIPv6 BUs, and,

—optionally, by the CN to prove ownership of its identifier or address (only if the CN itself uses an SUCV identifier or address).

The obtained security association (SA) could also be used for any other application of ESP. However, in the basic sucvP exchange, only the MN performs proof of ownership. Section 9.3.2 outlines the dangers this implies. Accordingly, general ESP usage should be limited to the extended sucvP exchange in which, in addition to the MN, the CN also uses SUCV for proof of ownership.

As for the choice of using AH or ESP to protect the BUs, we chose the latter. Given the benefits of integrity and data origin authentication inherent in the proof of ownership, we believe there is no added value in using AH to protect the IP headers of BUs once a SA has been established. This and the heated debate on the future of AH convinced us to use ESP.

sucvP is functionally independent of MIPv6, and is, in fact, a separate protocol. Proof of ownership in sucvP provides the authorization for the MIPv6 BUs, but the authentication is provided by IPsec ESP. These are two separate steps, which could run serially. For example, the sucvP step could be carried out over UDP (as our initial experimental implementation does), after which the ESP-authenticated BU could be sent.

However, for efficiency reasons, sucvP messages might contain MIPv6 BUs (along with sucvP3).

For sucvP to set up an IPsec SA (including an SPI) just in time to process an ESP header and its encapsulated BU, the sucvP payload is carried as an IP protocol number (currently unassigned). Furthermore, it must precede the ESP payload used to authenticate the BU.

6.1 Goals and Constraints

This design allows sucvP to satisfy these two objectives:

—not affect existing IPsec implementations more than absolutely necessary;

—support efficient BU processing by reducing as much as possible the number of round trips.

Furthermore, we assume there is no piggybacking with the BU, so no further payload follows.

sucvP has been designed based on the following considerations:

(1) the protocol should not rely on a third party (i.e., a global PKI, central KDC, etc.), although it could use one if available;

(2) not all nodes need to use SUCV addresses, only those that wish their BUs to be heeded (mobile nodes);

(3) not all nodes need to verify the validity of SUCV addresses, only those CNs that accept and handle BUs from MNs (these CNs must use SUCV as explained below to safely populate their binding caches);

(4) sucvP packets are exchanged directly between the mobile node and its correspondent nodes. They are not routed through the Home agent because the mobile node might be homeless or the home agent might be out of order

for a certain period of time. The implications for this decision are explored in Section 9.3.3.

6.2 Packet Exchanges

The proposed protocol that a mobile host uses to send a BU to its CN is the following:

—*sucvP1*: The MN sends a sucvP1 message (just to initiate the exchange) to its correspondent node. This message contains a *Nonce*, N1. This packet may contain a MIP *HomeAddress* Option containing the MNs home address. The CN might sometimes need the home address to decide whether it wants to pursue the protocol exchange or not. The source address of the packet is the MNs current CoA. Additionally, SUCV supports a very simple negotiation mechanism that works as follows: Optionally, the MN can express its desire to use certain Diffie-Hellman (DH) groups (for the ephemeral DH exchange), as well as algorithms for ESP authentication and for ESP encryption.

—*sucvP2*: The CN replies with a sucvP2 message that contains the following: N1, Client puzzle request, DH value ($g^y \bmod p$), Session_Key_lifetime. The CN may respond to any optional parameter negotiation included by the MN in sucvP1, by choosing those algorithms it wishes to support.

In order to defend against sucvP1 storms, a host might use the same DH value for a period of time. The sucvP2 contains a client puzzle to prevent DOS attacks [Aura et al. 2001]. Along these line, the CN may wish to ignore the optional negotiation of parameters initiated by the MN in sucvP1. In this case, the default algorithms (see Section 6.4) must be used by both parties.

When the MN receives sucvP2, it verifies that the nonce N1 is the same as what was sent in sucvP1. It then solves the puzzle. At this stage of the protocol, the MN:

(1) generates a DH value ($g^x \bmod p$) and derives from it and the DH received from the CN the session keys (see Section 6.3).
(2) computes skey_espauth (the ESP session key used to authenticate the MIPv6 BUs—see Section 6.3) lifetime as the minimum of the lifetime value suggested by the CN and its lifetime value.
(3) builds an IPsec SA. If ESP is used subsequently in the packet to secure a BU, the MN must use a fixed SPI assigned from the range 1 to 255 (currently unassigned).
(4) sends a sucvP3 packet. Note that this message is sent directly from the MNs CoA to the CN.

—*sucvP3*: A sucvP3 message contains the following fields: N1, Puzzle reply, Public key, and imprint; it has used to generate its HID, a DH value, the skey_espauth lifetime, and an SPI for the CN to use when sending BAs (secured via ESP) to the MN; this message must be signed by the MN with its private key (the public key is used to generate the HID).

Note that this sucvP3 might be followed by an ESP header authenticating an encapsulated BU. The authentication is performed using the SA available inline within this sucvP3 packet.

When the CN receives the sucvP3, it first checks for a valid Puzzle reply. It verifies the signature using the included Public key, and then verifies that this Public key and imprint produce the sucvHID used as part of the sender's address (as per Section 5.3. The CN can then conclude that the MN owns its the Home and CoA addresses.

At this point, the CN makes a note of this Public key and HID.

The CN can then compute the session keys (using the ephemeral DH value as described in Section 6.3). From the fixed SPI, the CN learns that the SA material is all inline in sucvP3. It proceeds to build an IPsec SA and processes this ESP header. In preparation for subsequent ESP processing of BUs, it computes an SPI and sends it in sucvP4. After this point, and thanks to this SPI, IPsec usage reverts to normal, that is, future BUs can be secured via ESP, unaccompanied by any inline sucvP material.

—*sucvP4*: In sucvP4, the CN sends an SPI. The MN will use this SPI in association with ESP in order to authenticate subsequent BUs. The CN authenticates sucvP4 with HMAC-SHA1 using the Session key (Skey_sucv) derived previously. Additionally, a CN that uses an SUCV address could sign sucvP4 instead. This possibility is explored below in Section 8.

A CN may include a BA along with sucvP4, and if so, it must use ESP for authentication. The SPI used is that communicated by the MN in sucvP3. When the MN receives a sucvP4, it must make note of the SPI corresponding to the CN.

As long as the MN uses the same HID interface identifier for its CoA, it does not have to prove the CoA ownership and BU authentication is enough.

Proving the CoA ownership can be very useful to prevent a malicious host from bombing a victim with packets by using the victim's address as CoA. For example, with "regular" Mobile IPv6, a host can initiate a large stream transmission from a server and then send a BU with the victim's address as CoA to the server. As a result, the stream will bombard the victim. If a host can prove that it owns its CoA, and that therefore it is not using another node's address as CoA, this attack can be avoided. However, this does not protect against a related attack in which the objective is not to bombard a particular host, but any given network prefix or link.

If for any reason the MN configures its CoA with a new interface identifier, it must restart the whole protocol sequence.

6.3 Deriving the Session Keys

We need to generate keying material and keys for the sucvP itself and for use with ESP.

$$\text{skeymat} = \text{prf}(\text{hash}(g^{xy} \bmod p), \text{N1}|\text{imprint}),$$

where N1 is the nonce used in sucvP1 and sucvP2.

6.3.1 *SUCV Session Key.*

$$\text{skey\_sucv} = \text{prf}(\text{skeymat}, g^{xy} \bmod p|\text{N1}|\text{imprint}|0).$$

Used with sucvP4 for authentication, and optionally with sucvP5 (see Section 8) for both authentication and encryption.

6.3.2 *Key for use with ESP.*

$$\text{skeymat\_espauth} = \text{prf}(\text{skeymat}, \text{skey\_sucv}|g^{xy} \bmod p|\text{N1}|\text{imprint}|1).$$

Used to authenticate BUs unaccompanied by SUCV packets (once sucvP is completed) as well as other applications of ESP (subject to the warning at the beginning of Section 6).

Note that whereas skey_sucv is the actual key used by the sucvP, skeymat_espauth is keying material used to derive the real key for use with ESP, that is, skey_espauth in an algorithm-specific manner.

6.4 Default Algorithms

The following algorithms are supported in our SUCV implementation:

—RSA for signing sucvP3.
—DH Oakley Group 1 [Orman 1998] for the ephemeral DH exchange.
—SHA-1 as the integrity algorithm.
—HMAC-SHA-1-96 [Madson and Glenn 1998] for ESP authentication.
—AES-128-CBC for sucvP5 and ESP encryption.

7. EXTENSION FOR CONSTRAINED DEVICES

In our sucvP protocol, a MN must:

(1) generate a DSA public/private key pair;
(2) sign the sucvP3 message;
(3) perform a DH exponentiation to derive the Skey.

All these operations are very computatively expensive especially if the MN is a constrained device (i.e., a PDA or a sensor with limited memory, battery, or CPU) [Modadugu et al. 2000]. Elliptic curve cryptographic algorithms might be more efficient but still too expensive to execute for a constrained device.

In this section, we propose an extension to our scheme for this type of constrained devices. Our goal is to off-load most of the expensive cryptographic operations of a MN to its HA. We assume that the MN and HA share a secret key, possibly obtained via *imprinting* [Stajano and Anderson 1999], and that the MN trusts its HA.

The proposed extension operates as follows:

(1) the HA generates the DSA keys (public and private keys) and sends the public Key to the MN via the secured channel.
(2) the SUCV id and HID is generated by the MN itself by choosing a k and computing sucvHID = prf64(hash(publicKey), k).
(3) when a MN wants to initiate a sucvP exchange with CN, it sends a SUCV_request messages, that contains the CN address and the k value, to its HA (authenticated with the shared key). The HA then initiates a

sucvP exchange with the CN. The HA then proves that it knows the private key corresponding to the public by signing the exchanged messages (sucvP has to be slightly modified here) and generates a session key, *SKey* using DH algorithm.

(4) The HA then sends the Skey to the MN via the secure channel.

(5) The MN can then send authentication BUs to the CN using the *SKey*.

With this extension all the expensive cryptographic operations are offloaded to the home agent but the session key that is used to authenticated the MIPv6 BU (*Skey*) is only known to the MN, its HA and the CN. A malicious host that wants to redirect a MNs traffic needs either to discover the HA–MN secret key or to find a public key/private key pair and a k' such that

$$\text{sucvHID} = \text{prf64}(\text{hash}(\text{public}), k').$$

Both are very difficult to achieve.

8. PRIVACY CONSIDERATIONS

A normal sucvP exchange consists of sucvP1 through sucvP3, and a subsequent sucvP4 authenticated using the session key. This basic protocol does not allow any hijacking attacks, so it already fulfills the security requirements for protecting BUs in MIPv6 as defined by the Mobile IP working group [Mankin et al. 2001].

8.1 Support for Random Addresses

A first concern regarding privacy is how to use random addresses as defined in RFC3041 [Narten and Draves 2001] in a mobile environment. The issue here is that, whereas these addresses hide a node's permanent identifier (perhaps derived from IEEE addresses), the node cannot prove address ownership of them so it cannot safely send BUs. This means that an MN cannot use RFC3041 addresses with route optimization. SUCV addresses are indistinguishable from those defined in RFC3041, with the added benefit that an MN can use them in a route optimized fashion. The basic sucvP outlined above in Section 6 already handles this case. The only consideration is that nodes interested in being anonymous may want to use *ephemeral* SUCV identifiers (as opposed to more permanent or longer-lived SUCV IDs) for this purpose.

Furthermore, if nodes wish to have higher protection against attackers than what is afforded by 63 bits in the sucvAddr, they can use an sucvID. The protocol exchange is the same, but since an sucvID is a pure identifier, as shown below, routing information must be included somewhere else in the packet, via home address options and routing headers (alternatively, tunneling headers could be used as well). This poses no difficulty if the MN operates as a client, always initiating contact with the CN, but would otherwise require mechanisms beyond the scope of this paper.

8.2 Support for Confidentiality

8.2.1 *Confidentiality.* If confidentiality is a concern, there is the possibility of an intruder in the middle gaining knowledge of the session keys, as explained in Section 9. In fact, sucvP prevents an intruder from impersonating a mobile node but not from impersonating a correspondent node. As a result, a MN might think that it is talking with its CN, whereas it is actually talking with an intruder. The MN may wish to make sure it is indeed talking to a given CN, whose address it has previously obtained (via, e.g., a DNS search, or a preconfigured list). If in addition to the MN, the CN also uses an SUCV address this problem can be prevented. We suggest that a CN uses a SUCV address when confidentiality is an issue and that the CN sign sucvP4 to prove its address ownership. By doing so, both MN and CN have the assurance that they are talking to each other and not to an intruder.

8.2.2 *Location Privacy.* In Mobile IPv6:

—each packet (BU and data) sent by a MN contains a HomeAddress option that reveals the MNs home address;

—each packet sent to a MN contains a routing header with the MNs home address.

As a result it is very easy for any host in the network to track the location of a MN by snooping its packets. If location privacy is an issue, a MN can use an ephemeral home address sucvADDR$_{\text{ephem}}$ instead of its actual one and only reveal its actual home address sucvADDR to its CN (see [Castelluccia and Dupont 2001] for more details). Packets (BU and data) sent over the network then use the ephemeral home address sucvADDR$_{\text{ephem}}$.

This privacy extension can actually be applied to our proposal. The MN will need an ephemeral SUCV identity sucvID$_{\text{ephem}}$, and defer revealing its more permanent SUCV identity sucvID after the CN has proven ownership of its address. This is accomplished roughly via the following extended protocol sequence:

—sucvP1: as usual

—sucvP2: the CN adds a bit to advertise its SUCV capabilities

—sucvP3: the MN proves ownership of its sucvADDR$_{\text{ephem}}$ (derived from an ephemeral public–private key). At this point, the MN derives session keys but is not yet sure it is sharing them with the CN itself.

—sucvP4: the CN proves ownership of its SUCV address by signing sucvP4 with its private key, at which point the MN knows the session keys have not been compromised by an intermediary.

—sucvP5: the MN uses the session key obtained above to send an encrypted payload revealing its actual SUCV Home Address sucvADDR. sucvP5 must be signed with the key used to generate the sucvADDR in order to prove its ownership.

Notice that if the MN wishes to use the stronger mode, it can do so by using an sucvID$_{\text{ephem}}$ and sucvID instead of sucvADDR$_{\text{ephem}}$ and sucvAddr, respectively.

As in the discussion above, this provides for more protection against attackers, with the proviso that, in practice, the MN may be limited to being a client. That is, it must initiate communication with the CN, because it is now using nonroutable entities (SUCV IDs versus SUCV Addresses).

9. SECURITY ANALYSIS

9.1 Hash ID Size Considerations

In SUCV addresses, one of the lower 64 bits is reserved as the local/universal bit (the u bit), so only 63 bits are actually usable as a hash.

Suppose the hash function produces an n-bit long output. If we are trying to find some input which will produce some target output value y, then since each output is equally likely we expect to have to try $2^{(n-1)}$ possible input values on average.

On the other hand, if we are worried about naturally occurring SUCV address duplications, then by the birthday paradox we would expect that after trying $1.2 \times 2^n/2$ possible input values we would have a 50% probability of collision [Menezes et al. 1997].

So if $n = 63$, you need a population of $1.2 \times 2^{31.5}$, that is, 3.64×10^9 nodes on average before any two produce duplicate addresses. This is acceptable especially if you consider that this collision is actually harmful only if the two hosts (that collide) are in the same site (i.e., they have the same 64-bit prefix), and have the same correspondent nodes. This is very unlikely. Additionally, assuming the collision is not deliberate the duplicate address detection (DAD) will detect it.

If an attacker wishes to impersonate a given SUCV address, it must attempt 2^{62} (i.e., approximately 4.8×10^{18}) tries to find a public key that hashes to this SUCV address. If the attacker can do 1 million hashes per second it will need 142,235 years. If the attacker can hash 1 billion hashes per second it will still need 142 years.

If we use SUCV Addresses as suggested in RFC3041 (perhaps renewing them as often as once every 24 h), an attacker would then have to hash 5.3×10^{13} hashes/second in order to be able to find a public key that hashes to the sucvHID of a given host.

Note that the previous analysis only considers the cost of computing the hash of the public key. Additionally, an attacker must also generate a valid (public, private) key pair. This is a significantly more expensive operation.

This would still leave open the possibility of brute-force attacks [Van Oorschot and Wiener 1994]. In this scenario, a bad guy, BG, could generate a huge table of PKs and their corresponding HIDs, assuming any fixed imprint. It could then look for matching real IP addresses. By doing so it would identify a victim for a hijacking attack. BG can send a BU to any CN without a binding entry for the victim's address (e.g., by targeting nonmobile fixed hosts as victims).

In general, such attacks are possible with hash functions, but not with *keyed* hash functions because they require interacting with the legitimate

user [Bellare et al. 1996]. Notice that normal usage of keyed hash functions requires an authenticated secret, which we do not have. Nevertheless, we can still limit exposure by creating the HID (or ID) using (in addition to the Public key) some key or known state that is established in advance of the sucvP interaction itself, and which will force interaction with the MN. This is the role of the imprint, sent by the MN to the CN in sucvP. Since the imprint is not authenticated, the CN could verify it independently of sucvP, perhaps by checking directly with the MN by routing it via the HA. True, the imprint is not a secret as expected for HMAC use, but it serves to severely limit which entities can launch the attack to only those entities with this privileged location, and within this time period. As another possibility, the imprint may instead be a quantity which the CN knows about the MN, and which the CN can verify independently using a separate subsystem (DNS, routing fabric, etc.). In this case, the attack is limited to only those nodes for which the imprint is also a valid quantity. Tying the HID in this manner may have undesirable consequences with regards to privacy and location independence (e.g., homeless operation).

Alternatively, one could always use sucvIDs (in which case the brute-force attacks would be nearly impossible).

Even for HIDs, actually carrying out such brute-force attacks remains highly unlikely in practice, and we claim our scheme remains secure even without requiring any of the above counter-measures.

9.2 Key Size Considerations

There are three ways that an attacker could break the MIPv6 security protocol presented in the paper:

(1) If an attacker find a DSA public/private key pair that hashes to the MN's sucvID, it can run a sucvP exchange with a CN and impersonate the MN. This can be achieved by a brute-force attack. The attacker tries several public keys as input to the hash function used to generate the sucvID. The difficulty of this attack depends on the size of the sucvID and is at least as hard as breaking a symmetric key algorithm that uses the same key size as the sucvID size (actually this is more difficult because the attacker must also generate valid public/private key pairs before performing the hash function).

(2) If an attacker can find the public/private key pair that is used to generate the sucvId and sign sucvP3, an attacker can impersonate a MN in sucvP. Breaking a DSA system depends on the DSA modulus and subgroup.

(3) If an attacker can retrieve the generated session key it can send fake BUs on behalf of the MN and redirect its traffic. An attacker has two ways of retrieving the session key: (1) generate it from the DH values exchanged between the MN and the CN, or (2) perform a brute-force attack on the session key itself. The difficulty of these attacks depends respectively on the DH modulus size and the session Key size.

A security system is consistent if all the components of the security chain provide the same security levels and none of them is a weak link.

Most of the security parameters used in our proposal (DH modulus size, Session key size, DSA subgroup) can be adjusted. The only fixed parameter is the SUCV identifier itself. It is either 63 bits long (i.e., we use an sucvHID) or 125 bits long (if using an sucvID itself).

If we use sucvHIDs, the security of our proposal depends on these 63 bits. Accordingly, the symmetric key strength should not be less, not would we gain much by it being significantly stronger. In light of Orman and Hoffman [2004], Oakley group 1 is about enough for this application (although there are other more conservative views [Lenstra and Verheul 1999]).

However, if we use suvcIDs, we will need a symmetric key strength of almost 128 bits (125 bits) of output from our *prf*. Notice that 96 bits symmetric keys are generally considered safe for another 20 years or so. However, if we want to keep up with the strength afforded by the sucvID itself, we would need to use other MODP groups [Kivinen and Kojo 2003]. For example, MODP group 5 with exponents of 1563 bits should be enough to derive 90 bit symmetric keys. MODP group 6 with 2048 bits should be used to produce 100 bit symmetric keys.

9.3 Intruder-in-the-Middle Attacks

As described in Section 6, a mobile node and its correspondent node derive a shared (symmetric) key to authenticate the MIPv6 BUs sent by the MN.

The MN and its CN derive the shared key using DH algorithm.

—The CN chooses a random secret y and sends $g^y \bmod p$ to the MN (in the DH value field of sucvP2)

—The MN chooses a random secret x and sends $g^x \bmod p$ to its CN (in the DH value field sucvP3)

The session key shared by the MN and its CN is then a hash digest of $g^{xy} \bmod p$ (g and p are known by the MN and CN).

9.3.1 *Summary of the Attack.* DH is known to be vulnerable to the *intruder-in-the-middle* attack on unauthenticated DH key agreement:

$$\text{CN} \longrightarrow g^y \longrightarrow \text{Intruder} \longrightarrow g^{y_i} \longrightarrow \text{MN}$$
$$\text{CN} \longleftarrow g^{x_i} \longleftarrow \text{Intruder} \longleftarrow g^x \longleftarrow \text{MN}$$

The intruder intercepts g^y sent by the CN and sends g^{y_i} to the MN. The intruder also intercepts g^x sent by the MN and sends g^{x_i} to the CN. As a result, MN shares the key g^{xy_i} with the intruder (it actually thinks that it is sharing this key with its CN). The CN shares the key g^{x_iy} with the intruder (it actually thinks that it is sharing this key with the MN). The Intruder can then impersonate the MN and the CN.

In our protocol, the MN signs sucvP3 (with contains g^x). As a result, the intruder cannot modify nor replace this message. This only thing that the intruder

could do is the following attack:

$$\begin{aligned}
\text{sucvP1:}\quad & \text{CN} \longleftarrow \text{HID}' \longleftarrow \text{Intruder} \longleftarrow \text{HID} \longleftarrow \text{MN}\\
\text{sucvP2:}\quad & \text{CN} \longrightarrow g^{y} \longrightarrow \text{Intruder} \longrightarrow g^{y_i} \longrightarrow \text{MN}\\
\text{sucvP3:}\quad & \text{CN} \longleftarrow g^{x_i} \longleftarrow \text{Intruder} \longleftarrow g^{x} \longleftarrow \text{MN}
\end{aligned}$$

In sucvP1, MN sends its HID by virtue of sending from its address (the HID is just the bottom 64 bits in the address). The intruder could replace this HID by another value, say HID_i, without affecting return routability, as long as the prefix remains the same. In sucvP2, the CN sends its DH value g^{y}, which is replaced by the intruder for g^{y_i}. In sucvP3, the MN sends its g^{x}. Notice that the intruder can replace it by another g^{x_i} as long as this g_i^x is used to create HID_i.

9.3.2 *Risks.* The keys created are derived from g^{xy_i} (between the MN and the intruder) and g^{yx_i} (between the intruder and the CN).

So the intruder cannot pass itself off as MN (assuming it is computationally unfeasible to find another private–public pair that generates the same HID). It can, however, pass itself off as MN_i, where this is the address formed from HID_i. This means that it is not possible for an intruder to hijack an existing communication between MN and CN. But if the intruder is present at the very beginning of the communication, and if it sits on the path it could supplant MN. In so doing it could obtain knowledge of any session keys derived for this communication.

If the session supported encryption, the endpoints might be led to believe in the privacy of their conversation, oblivious to the fact that the intruder could snoop. For example, suppose an MN established an sucvP session with an CN. Subsequently, and using this optimized path, an application (e.g., telnet) started. If a security policy database required all such application traffic to be encrypted, a misconfigured system might leverage the existing sucvP session and use ESP for confidentiality. This would result in the intermediary being privy to all the application traffic.

Because of this, sucvP session keys must not be used for anything more than securing BUs. In other words, IPsec traffic selectors in the SPD must limit use of SAs obtained via sucvP for the sole purpose of securing BUs. In order to avoid any potential misapplication of these SA's BUs must not be piggybacked.

Not heeding the above guidelines may result in the aforementioned snooping attack. Nevertheless, the attacker would have to remain on the path forever. This interception is possible because of the nonauthenticated nature of the example. Of course, if the exchange is authenticated, perhaps as contemplated by default by HIP [Moskowitz 2001, 2003, 2004], this would not be possible. Even if this interception is possible, once the intruder ceases to be on the path between MN and CN there is nothing further it can do. In other words, the use of unauthenticated SUCV entities does not add any risk to those that currently exist. Even unauthenticated SUCV, eliminates the possibility of on the path redirection of traffic. Notice that with current MIPv6, "off the path" (as well as "on the path") redirection of traffic is possible.

In some case, a MN might request to its CN to acknowledge the reception of the BU. The intruder could actually fool the MN by sending an acknowledgement with the CN address as source address (note that the intruder could also authenticate this acknowledgment, since it knows the key used by the MN, g^{xy}). This might confuse the MN that has received an acknowledgement but keeps receiving the packets from the CN via its home agent (note that the same problem exists also will current Mobile IPv6 specification)!

One solution to these problems is for the CN to use an SUCV address and to sign sucvP2 (the message that contains the DH value). Then, the intruder will not be able to substitute g^y by g^{y_i}.

Of course, the intruder can hinder successful completion of the sucvP, thus preventing the CN from heeding the MN's BU using route optimization to the MN. In effect, this is a DOS attack against route optimization, and it leads to service degradation not disruption.

The previous security analysis shows that the protocol described in Section 6 prevents any intruders from redirecting the traffic addressed to a mobile host's home address and consequently provides the minimal Mobile IP security requirement [Mankin et al. 2001].

9.3.3 *Should sucvP Involve the Home Agent?.* This section examines two ways of involving the home agent in the sucvP protocol to secure BUs:

(1) Using a home address option, and
(2) Introducing a home test.

These issues are examined in more detail elsewhere [Nikander et al. 2003a, 2003b] so here we will summarize briefly.

First, what if we assume sucvP1 was carried with a home address option, and then sucvP2 traveled via the home agent? At this point, the home agent can check that the validity of this MN_i (corresponding to HID_i), its current care-of address, and so on. In this case, none of the above snooping would be possible. In order to further mitigate the sucvP2 packet from being redirected, the MN must check upon its reception that it was sent tunneled by its home agent. Home address options can be misused to setup distributed DOS attacks whereby these options are sent to numerous hosts prompting them all to respond to the same address. Even if CNs exercise caution when sending their sucvP2 packets as instructed via a home address option, the nature of DDOS attacks is such that any given CN may not send more than a few sucvP2s to the same home address region (same prefix), the collection of thousands of such responses may be sufficient to clog a target network. In short, unauthenticated home address options can be used to launch reflection attacks.

Another way to involve the home agent is to perform the so-called *home test* [Johnson et al. 2003]. Not doing so allows a residual vulnerability known as a *home flooding* attack. The attacker creates an SUCV address with the prefix from a target link. It visits the target link at one point, initiates a packet exchange with a CN, and subsequently uses sucvP to create a binding entry at the CN (it may be possible, though unlikely for the MN to create the binding entry with a CN as part of the initial packet exchange). Once this binding entry

is in place, the malicious MN causes a large packet stream to be sent from the CN. Once the binding entry disappears at the CN (either through natural expiration or by the MNs deleting it, this voluminous packet stream becomes a DOS attack on the target link.

The above analysis shows the pros and cons of using the home address option. Whereas unauthenticated home address options should not be heeded, a real deployable protocol should protect against the home flooding attack by carrying out both a home test as well as a care-of test. Nevertheless, for illustrative purposes we have decided to show a more simplified protocol in sucvP by exchanging packets directly between the MN and the CN. At any rate, it seems that an SUCV-based route optimization scheme would only need to carry out the home test once (so the CN can verify that this is, in fact, a node authorized to be a MN within the link in question). Once this is resolved, subsequent BUs to the same CN would not necessarily have to repeat the home test, proceeding along the lines of the sucvP protocol we have described.

9.4 DOS Attacks

DOS attacks that exhaust a host resource (memory and computational resources) are a major security threat on the Internet. In the section we study the behaviors of the protocol described in Section 6 against DOS attacks.

—*sucvP1 storm*: Malicious hosts could try to attack a host by sending a storm of sucvP1 messages. We prevent this potential attack as follows:
  (1) when receiving a sucvP1, a host does not create any state and replies with a constant message (sucvP2) that contains a client puzzle [Aura et al. 2001].
  (2) An host only creates state if it receives a valid puzzle reply to its puzzle request (in sucvP3).

—*sucvP2 storm*: Malicious host could try to attack a host by sending a storm of sucvP2 messages. We prevent this attack by inserting a nonce, N1, in the sucvP1. If a host receives a sucvP2 with a nonce N1 that is not equal to the nonce N1 that it has set in the initial sucvP1, this sucvP2 must be rejected.

  Note that an intruder (between the MN and its CN) could intercept the sucvP1 and reply to the MN with a fake sucvP2 containing a valid N1 and an intentionally difficult puzzle request. The MN would then spend a lot of CPU and power computing the puzzle reply. This attack can be avoided if the MN had a mean to authenticate the address used by its CN. One solution is that the CN uses a SUCV address and signs sucvP2.

  Instead of this heavy alternative, we suggest that a MN simply reject any sucvP2 messages that contain an overly complex client puzzle request. Of course, the MN itself defines the complexity threshold of the puzzle request as a function of its processing power.

  As a result, the attack that consists of sending complex puzzles (in sucvP2) to a MN, in order to exhaust its computing resources, will not be successful, because the MN will drop the sucvP2. The MN service will be degraded

(because its incoming packets will then be routed through its home agent) but not disrupted.

—*sucvP3 storm*: Malicious hosts could try to attack a host by sending a storm of sucvP3 messages. We prevent this attack by using a client puzzle. A host accepts a sucvP3 message only after verifying that the puzzle reply (contained in the sucvP3) is valid.

10. OTHER APPLICATIONS

The previous sections have explained in some detail how the SUCV properties can be used for one particular application: Mobile IPv6. Nevertheless, there are many other applications of such properties and of CBIDs in securing IPv6 networking, as well as in other areas such as peer-to-peer or application level security. This section discusses briefly several of these other applications. Other applications which are not discussed include securing neighbor discovery [Arkko et al. 2002; Aura 2004; Montenegro et al. 2003], opportunistic encryption [Castelluccia and Montenegro 2002a] and securing internet storage facilities [Bassi et al. 2003], for example.

10.1 Securing Identities in Ad Hoc and Peer-to-Peer Networks

Ad hoc or peer-to-peer networks (called *impromptu* networks henceforth) pose many problems with respect to securing their highly dynamic structures. A naive approach assumes any given node can trust all other nodes in the impromptu network for any type of operation (e.g., engaging in some cooperative and perhaps confidential activity). We improve on previous efforts to secure group authorization (including membership) by employing CBIDs for node and group identification, and then use these in authorization certificates. These allow groups (or nodes) to authorize nodes (or other groups). We have employed a similar approach for Multicast and Anycast group membership security in IPv6 [Castelluccia and Montenegro 2003].

Our approach enables highly flexible and robust impromptu security services in an inherently distributed fashion. Previous work on securing impromptu networks has assumed the existence of a traditional PKI, of some web of trust or of some mechanism to distribute keys and shared secrets. We believe these assumptions are unrealistic in impromptu networks.

10.1.1 *Secure Node Identity.* First of all, we must start by defining an addressing model. Previous efforts for ad hoc networks conclude that since there is no aggregation to the degree possible with regular fixed networks, addressing can be more flexible. Thus, we propose to use CBIDs, that is, pure identifiers, with no topological meaning.

A node autoconfigures its (crypto-based) identifier (CBID) as explained above. Given the secure correspondence between identity and public key, the latter can be communicated by the node itself to its peers or group managers. This simplifies key management, since no third parties need to be involved either in creating or distributing the public keys. However, the node can prove ownership of the CBID by signing packets with the corresponding private key.

Any other node can verify the signature without relying on any centralized security service such as a PKI or KDC.

These characteristics (1) make CBIDs a very scalable naming system, well adapted to ad hoc environments, and (2) provide an autoconfigurable and solid foundation for nodes to engage in verifiable exchanges with each other.

10.1.2 *Dynamic and Secure Node Authentication.* The first application of the above is to protect basic exchanges between two peers or nodes in a network from malicious intermediate hosts. For example, in on-demand ad hoc routing protocols (e.g. [Perkins et al. 2002, 2003]), nodes discover each other by exchanging "route request" and "route reply" messages. We have shown how CBID can protect this basic exchange from impersonation attacks [Castelluccia and Montenegro 2002b]. Similar work has been done for the JXTA open-source peer-to-peer protocol [JXTA n.d.] by extending its "PeerID" definition to accommodate crypto-based PeerIDs [Crypto-ID JXTA n.d.].

10.2 Dynamic and Secure Group Membership

Beyond securing identities, the SUCV properties of CBIDs lend themselves to an even more powerful construct by using them as *issuers* of authorizations whose beneficiaries (or *subjects*) are other CBIDs. Such mechanism allows us to complement the *Default Trust Rule* 4 with explicit and secure authorizations. These can be expressed via authorization certificates, similar to how they are used to *Secure Group Management for IPv6* [Castelluccia and Montenegro 2003]. The expressiveness of authorization certificates and the self-authenticating nature of CBIDs enable a decentralized and highly dynamic group management facility.

Authorization certificates have the following form:

$$\text{Cert} = (\text{group, node, delegation, tag, validity})$$

In the above, *group* is a group CBID for the entire impromptu network, or for a subset of it. Appropriately, the certificate is signed with the private key that corresponds to it.

Here, *node* is the CBID of the beneficiary of this authorization, that is, the node that is authorized by the group to join it or perform certain services on its behalf.

delegation is a Boolean (in, e.g., SPKI [Ellison et al. 1999] or KeyNote2 [Blaze et al. 1999]) that specifies whether or not the group has allowed the *node* to further delegate the permission expressed in the next field.

tag is the authorization to be a member of the signing group, or to perform certain services as authorized by the group.

This is an example of how a single node A with CBID of $A_$CBID could start an ad hoc network (really a group within a perhaps already physically existing ad hoc network) by following these steps:

—A creates the group public and private key pair: $G_$PK and $G_$SK.

—A creates the group identifier: $G_$CBID = hash($G_$PK).

—A as the *group controller* issues a certificate to allow itself into the group:

(G_CBID, A_CBID, true, "groupMembership", someDuration)

—A as the *group controller* admits another node (e.g., B) into the group by issuing the corresponding certificate:

(G_CBID, B_CBID, false, "groupMembership", someDuration)

Now, either A or B can prove to other nodes that they are legitimate members of the group by

(1) sending a message which includes their certificate, and,
(2) signing it with their private key (A_SK or B_SK, respectively).

11. USER-FRIENDLY PUBLIC KEY EXCHANGE

Even though the use of SUCV identifiers and addresses solves the identifier or address ownership problem, it is also important to be able to tie a first identity known from some context to an identity used in the SUCV context. Typically, this is done via a public key infrastructure, in which a *distinguished name* constitutes the first identity, as vouched for by some certificate authority. This has the drawbacks that (1) one must trust that the certificate authority has not made a mistake (perhaps unknowingly because of a collision in the distinguished name or due to some foul play somewhere in the process), and (2) it is hard to reconcile such a rigid approach with impromptu, ad hoc styles of computing. For example, two users that know each other personally should be able to reap the benefits of this knowledge in order to bootstrap some network-level security between their personal devices.

Let us suppose Alice knows Bob (they are friends, colleagues, etc.). Furthermore, they both have trusted and *exclusively personal* devices: Alice has A and Bob, B. Furthermore, the devices have mutual reachability by virtue of being in the same network. Since human beings are notoriously inadequate for data transfer, it is not practical for Bob to either display or dictate his certificate (or raw Public Key) to Alice, in order for her to enter this long and unwieldy string of bits.

Humans are quite adept, however, at certain types of pattern recognition. The basic idea is to use humans for what they are good at: (1) pattern recognition and (2) authenticating (recognizing) other humans (e.g., via auditive or visual information); while leveraging computers for what they are good at: exchanging data at high speeds over a network.

In the past, a similar approach has been taken assuming the devices have a bitmapped display [Perrig and Song 1999]. However, in order to permit bootstrapping of security information in as many cases as possible, it is important to not rely on bit-mapped graphics and to reduce the display requirements to a minimum. Accordingly we have chosen to use a human-friendly rendering based on the One-Time Password (OTP) dictionary [Haller and Metz 1996]. This dictionary includes 2048 (2^{11}) simple English words. Depending on its position within the dictionary, each such word can replace 11 bits in a bit string. Once translated into a sequence of OTP words, such a *sentence* can be readily displayed or read.

A summary of our approach [Bailly 2002] follows:

Alice recognizes Bob using an authentic (but not necessarily secret) alternate channel. This channel could be the audiovisual contact they have by virtue of being one in front of the other, or a phone line which can allow Alice to recognize Bob's voice, or perhaps even an email channel which has been deemed for other reasons and previous usage to be authentic. Alice wishes to use this alternate authentic channel she shares with Bob in order to bootstrap a secure channel between their devices. Alice makes her device A create a very simple message meant to elicit Bob's device B to respond with its CBID (i.e., either its SUCV ID or its SUCV Address, whichever may be applicable): B_CBID. Notice that in order to avoid DOS attacks, Bob's device is not necessarily enabled at all times to respond to such solicitations. This could happen as a result of Alice asking Bob to enter into such an exchange with her.

Alice's device A then sends the request to the network. Since A ignores B_CBID (learning it is the whole point of the exchange), it uses broadcast, multicast or any other means to reach B without explicitly addressing it. In its simplest form, the protocol is asymmetric: Alice authenticates Bob's device. It can be modified with symmetry in mind, or it can simply be rerun in the other direction. At the end, the entities have all that is required in order to then bootstrap a secure channel by running the *sucvP* protocol as shown previously 6. Nevertheless, a symmetric protocol is useful as it allows for Bob to be an embedded device (e.g., a wireless access point in a public location) as explained below.

The following exchange happens on the channel available between the devices:

—A: In its simplest form, the protocol starts by A broadcasting a request for CBIDs.

$$A \longrightarrow \text{Broadcast} : \boxed{\text{RequestCBID}}$$

If Alice includes her own public key (or certificate) used it used to derive A_CBID, the protocol can be rendered symmetric by Bob's authenticating Alice's device later on.

Of course, if Bob is an embedded device, the simple variant in which it responds to requests works best.

Another potential addition is a nonce, if Alice wants to guarantee freshness in Bob's subsequent reply.

Presumably, the source address of this packet is equivalent to A_CBID. For example, in IPv6, it would be either an SUCV identifier or address, whereas in JXTA it would be the crypto-based *PeerID*.

—B: B responds to the originating address by sending the public key (or certificate) it used to generate B_CBID.

$$B \longrightarrow A : \boxed{\text{ReplyCBID, Public Key of B}}$$

B does not have to explicitly include B_CBID as it is easily calculated from the material included in the message. B may, however, optionally sign the

message, perhaps including any nonce that *A* may have included in its request message. This would provide assurance to *A* that the message is fresh and not a replay of a previous message. However, in its simplest form, *B* is communicating a static data (its public key), so this may be overkill.

Notice that potentially many devices (not just Bob's) may hear the request and respond to it.

—*A*: If a signature is included *A* verifies it, at which point it knows for sure that the message was sent by a node whose public key is as included in the message.

It translates *B*_CBID into a user-friendly sequence of OTP words known as a *sentence*. A minimum of, say, eight words would be enough as it provides at least 88 bits of the hash of the public key. The device then makes this OTP sentence available to its user Alice.

This part of the exchange happens in the authentic (but not secret) channel between Alice and Bob:

—At this point, Alice has Bob's sentence which consists of, say, at least eight words. The last step is carried out in the Alice asks Bob for his sentence and verifies that it exactly matches what her device *A* received via the network (i.e., the device channel). If it matches, Alice has succeeded in authenticating the data as coming from *B*, Bob's device. She can now effect the encapsulation of such knowledge by the creation of a corresponding certificate in her device *A*. Similarly, Alice and Bob can check the sentence derived from *A*_CBID Bob thus reversing the protocol.

—If Bob is an embedded device, the sentence representation of its *B*_CBID could be constantly on display. The device would simply respond on the network channel to requests and users would then check that the sentence representations matched.

12. RELATED WORK

CAM [O'Shea and Roe 2001] presents a solution to the Mobile IPv6 security problem that is very similar to our proposal. When we first submitted our work to the IETF in April 2001, we were unaware of this work. CAM also uses IPv6 addresses derived from cryptographic keys to solve the MIPv6 address ownership problem. The main differences between CAM and SUCV are:

—CAM relies on signatures to authenticate BUs. In SUCV, a signature is only used by the sucvP protocol to prove address ownership. A session key is derived between the MN and its CN, after which the BUs are authenticated using IPsec.

—CAM requires that the CN and MN have a synchronized clock to protect against replay attacks. We believe that this is a strong assumption that is not always practical. SUCV uses a puzzle mechanism to protect against such attacks.

—CAM only uses addresses derived from cryptographic keys. In addition, SUCV defines the concept of an SUCV identifier that is longer (128 bits) and therefore more secure. As explained previously, an SUCV identifier may be used as a nonroutable Home Address when the mobile node is the client (i.e., when it initiates the communication), or when routing information for this identifier is otherwise obtainable.
—In addition to proposing a mechanism to solve the address ownership problem, SUCV also provides provide data and location privacy.
—Digital signatures are very expensive operations that cannot be performed on small mobile devices such as PDA or sensors. SUCV proposes an extension for constrained devices that off-load all of the expensive computations (signature, DH exponentiation, and session key generation) to the home agent, while still providing end-to-end security (see Section 7).
—CAM uses a hash to derive what we call the sucvHID. We use a prf-based mechanism.

The BAKE proposal [Nikander and Perkins 2002] presents a solution to the Mobile IPv6 security and key distribution problems. As compared to SUCV, BAKE is lighter in terms of computational overhead, but weaker security-wise. In fact, BAKE only requires a few hash operations but is subject to man-in-the-middle attacks if the attacker resides along the path between the CN and the MN's home agent. The BAKE protocol uses three messages. The first one is a trigger sent by the MN. The CN replies with a second message that contains a cryptographic token. This message is sent to the MN via its HA. Upon reception of the message, the MN sends a third message to the CN that contains another cryptographic token. The generated key is derived from the two tokens. As a result, only an intruder that can hear messages 2 and 3 could reconstruct the session key. Also, since message 2 is sent via the home agent, the CN has reasonable assurance that the home address belongs to the MN. Being lightweight and easily deployable, BAKE has its benefits. Nevertheless believe that ultimately a more secure solution must be adopted. An intruder that is close to the CN (e.g., on the same wireless link) can hear all three messages and be a potential attacker.

13. CONCLUSION

We propose a protocol for a mobile node to prove the ownership of its addresses and to authenticate the BU that it sends to its CN. This protocol was made part of Mobile IPv6 for deployment reasons. However, the address ownership problem is more general than Mobile IPv6 and other protocols and applications might need this functionality. The sucvP protocol, in fact, can be used by all protocols and applications above it. Communicating hosts can use it to prove to each other that they own their respective addresses. They can further use it to derive shared keys that can be used by the hosts' protocols and applications. This protocol provides mutual ownership proof (i.e., proves the address ownership of both hosts) and/or unilateral ownership proof (i.e., proves only

the address ownership of one of the hosts). We have also shown how the addition of expressive authorization certificates results in a decentralized and very dynamic authorization mechanism that is a powerful new tool in networking security.

APPENDIX

Implementation Considerations: sucvP Protocol Specification and Packet Formats

This section presents some of the design choices made in our sucvP prototype implementation.

sucvP uses a very simple negotiation. First of all, hopefully no negotiation of cryptographic parameters is necessary, as the defaults should be widely applicable. Any departure from the default is proposed by the initiator using TLV encoding and either accepted (and used) or rejected by the responder.

Furthermore, sucvP deals with cryptographic suites, similar to TLS and JFK [Aiello and Bellovin 2002] usage.

The initial set of algorithms that MUST be supported is shown below

Suite	Identifier
SUCV_RSA_WITH_ESP_AES_128_CBC_HMAC_SHA1	{0×00,0×01}
SUCV_RSA_WITH_ESP_3DES_CBC_HMAC_SHA1	{0×00,0×02}
SUCV_RSA_WITH_ESP_NULL_HMAC_SHA1	{0×00,0×03}

Notice that SUCV only supports RSA certificates, ESP headers, and SHA1. These are repeated in the names of the suites above for clarity, not to imply that departures are allowed (to DSS, AH, or MD5, for example).

The default suite that requires no negotiation is

SUCV_RSA_WITH_ESP_AES_128_CBC_HMAC_SHA1

This suite has been assigned an identifier, but the identifier will never be used in an SUCV payload because fixed payloads are enough to support the default case. sucvP does not allow individualized negotiation for transform types 1 (encryption algorithm), 2 (pseudo-random function), 3 (authentication type), and 4 (integrity algorithm). By using the above default suite, SUCV mandates the following: AES_128_CBC for encryption, HMAC_SHA1 for pseudo-random function, RSA for authentication type and SHA1 for integrity algorithm.

It does, however, allow for negotiation on DH group (IKEv2's transform type 5). In doing so it reuses the numbering used in IKEv2 [Kaufman 2004]. The mandatory group for DH exchange is group 5 (1536 bit MODP) and RSA signature uses exponent 65,537.

Packet Formats

sucvP is an IPv6 protocol, identified by protocol number (sucvP TBD by IANA) in the Next Header field of the immediately preceding header.

The Next Header field identifies the next protocol in the IPv6 daisy-chain header as per normal IPv6 usage.

The Version field for this version of sucvP MUST always contain the number 1.

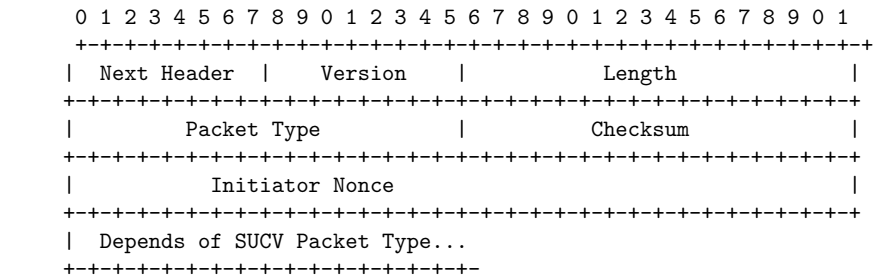
The Length field indicates the length in bytes of the whole sucvP header, including any possible options.

The Checksum is the 16-bit 1's complement of the 1's complement sum of the entire sucvP message starting with the Next Header field (see below), prepended with a "pseudo-header" of IPv6 header fields, as specified in Section 8.1 of Deering and Hinden [1998]. The Next Header value used in the pseudo-header is (sucvP TBD).

For computing the checksum, the checksum field is set to zero.

The Packet Type is defined to be {packet number|sequence type}; the packet number is related to the relative position within the exchange, while the sequence type is related to the sort of SUCV exchange we wish to perform. For example, an sucvP3 packet used to establish initiator's proof of ownership without privacy considerations would be labeled with the packet type $\{0 \times 03|0 \times 01\}$. It would be labeled $\{0 \times 03|0 \times 02\}$ if it is used to establish mutual proof of ownership or to avoid disclosure of identity. On the other hand, an sucvP1 packet would always be labeled $\{0 \times 01|0 \times 01\}$, as there is only one type of p1 in both exchange types {initiator|responder|privacy}.

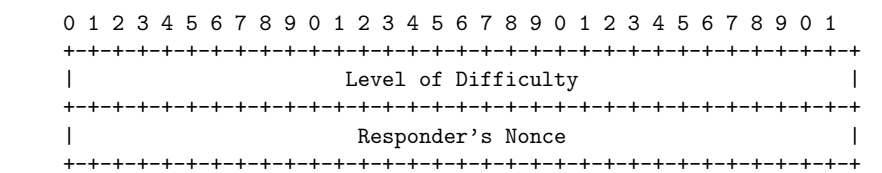
Common Header Format



Cookie Puzzle Request Format

Level of Difficulty is k

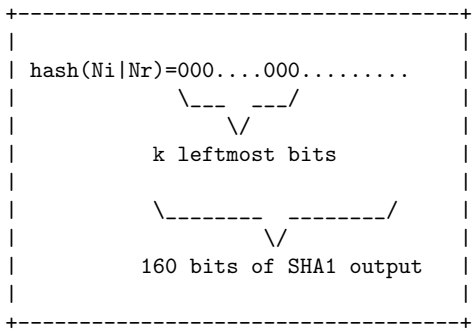
Responder's Nonce is Nr

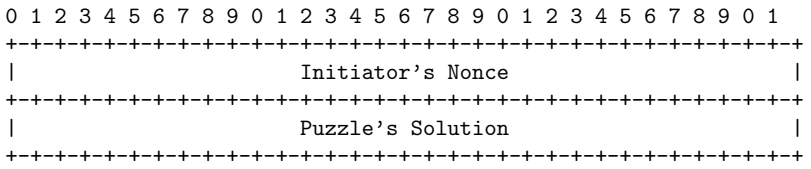


Cookie Puzzle Reply Format

Initiator's Nonce is Ni

Puzzle's Solution is an integer X which satisfies the property :

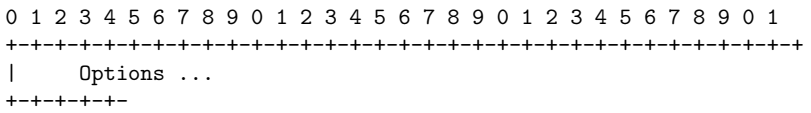




P1 Packet Format

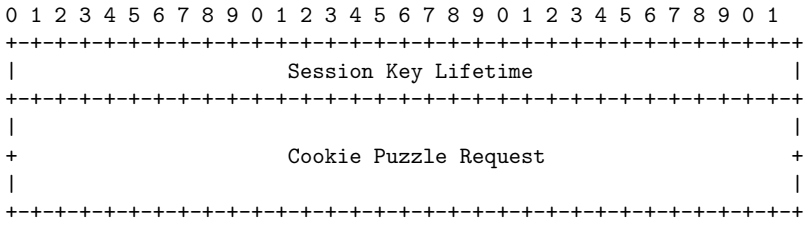
Type 0×0101

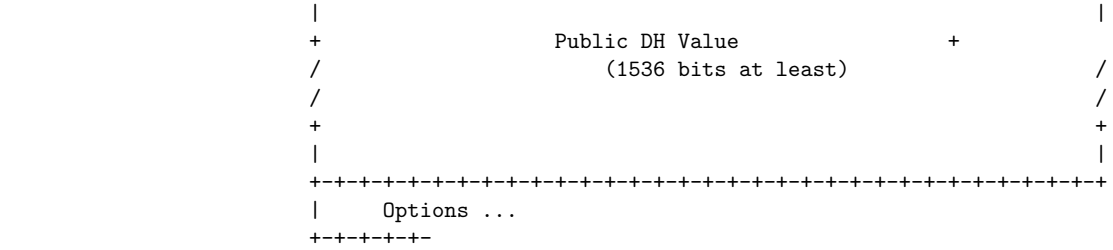
P1 is just the common packet format without additional fields, although a vendor-specific or application-specific options field is possible.



P2 Packet Format

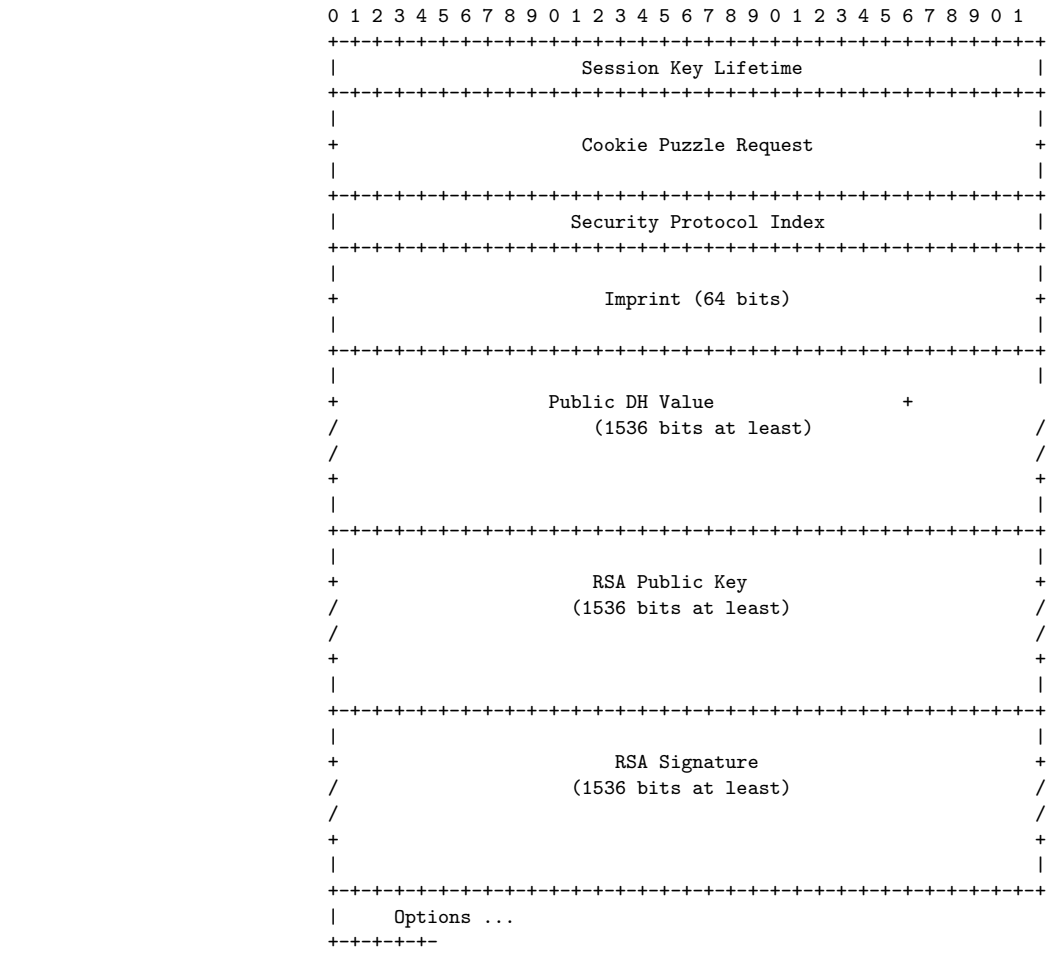
Type 0×0201





P3 Packet Format

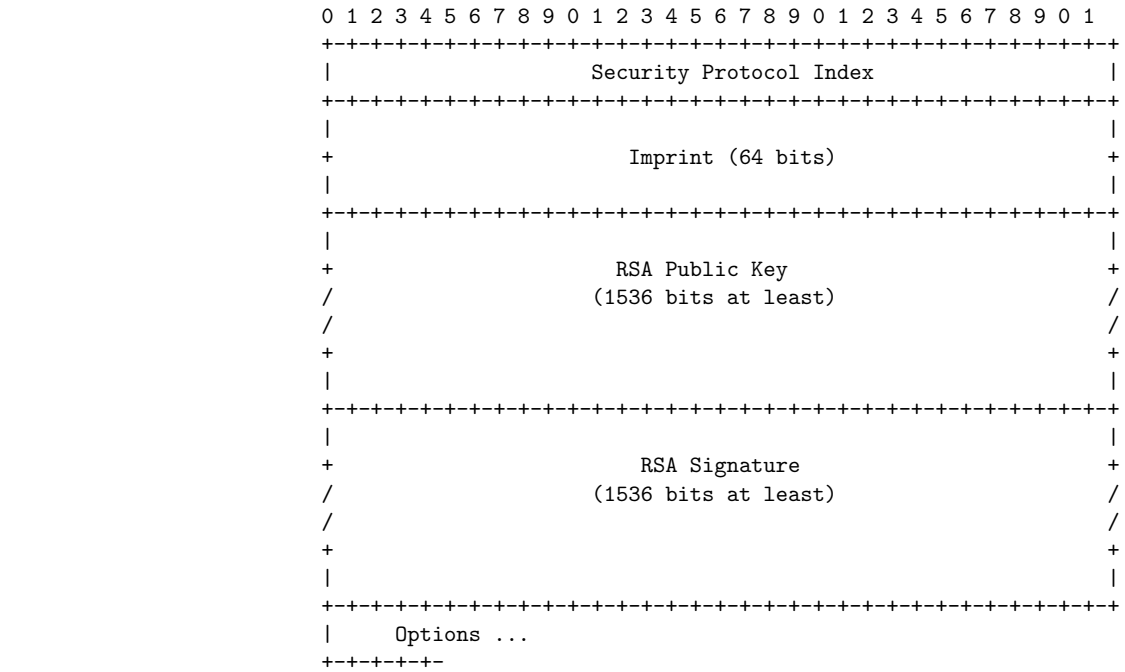
Type 0×0301



P4 Packet Format

Type 0 × 0401 (to be used to prove only initiator's ownership)

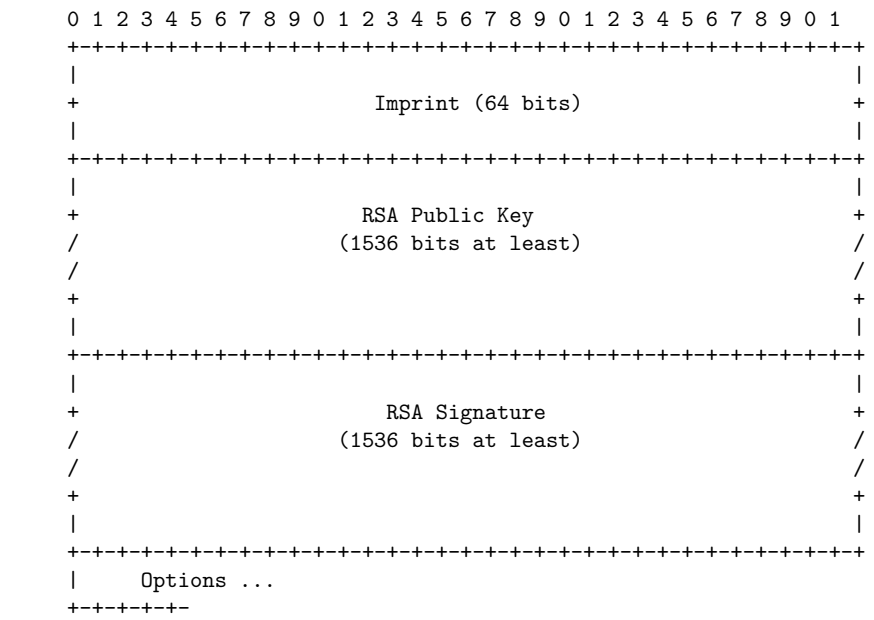
Type 0 × 0402 (to be used to prove responder's address ownership)



P5 Packet Format

Type 0 × 0502 (to be used to protect disclosure of initiator's identity)

Note that this packet must be protected within an ESP payload, to avoid disclosure of initiator identity (Public Key).



ACKNOWLEDGMENTS

The authors appreciate the helpful comments received from the following individuals: Erik Nordmark, Alberto Escudero, Lars Henrik Petander, Imad Aad, Pars Mutaf, Damien Bailly and the anonymous reviewers. Special thanks to Julien Laganier who helped us with the packet formats.

REFERENCES

AIELLO, W., BELLOVIN, S. M., BLAZE M., CANETTI, R., IOANNIDIS, J., KEROMYTIS, A. D., AND REINGOLD, O. 2002. *Just Fast Keying (JFK)*. IETF, draft-ietf-ipsec-jfk-04.txt, work in progress.

ARKKO, J., AURA, T., KEMPF, J., MANTYLA, V.-M., NIKANDER, P., AND ROE, M. 2002. Securing IPv6 neighbor discovery and router discovery. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe)*. ACM Press, Atlanta, GA, USA, 77–86.

AURA, T. 2004. *Cryptographically Generated Addresses (CGA)*. IETF, draft-ietf-send-cga-05.txt, work in progress.

AURA, T., NIKANDER, P., AND LEIWO, J. 2001. DOS-resistant authentication with client puzzles. In *Proceedings of the Security Protocols Workshop 2000*. Cambridge, UK, April 2000. Lecture Notes in Computer Science, vol. 2133. Springer, Berlin, 170–181.

BAILLY, D. 2002. *Cbjx: Crypto-based jxta* (an internship report) *6*, 3 (July 2004), 108–109.

BASSI, A., BECK, M., LAGANIER, J., AND PAOLLINI, G. 2003. Towards an ipv6-based security framework for distributed storage resources. In *CMS 2003 Seventh IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*.

BELLARE, M., CANETTI, R., AND KRAWCZYK, H. 1996. Message authentication using hash functions—the HMAC construction. *RSA CryptoBytes 2*, 1.

BLAZE, M., FEIGENBAUM, J., IOANNIDIS, J., AND KEROMYTIS, A. 1999. *The KeyNote Trust-Management System Version 2*. IETF, RFC2704.

CASTELLUCCIA, C. AND DUPONT, F. 2001. *A Simple Privacy Extension for Mobile IPv6*. IETF, draft-castelluccia-mobileip-privacy-00.txt, work in progress.

CASTELLUCCIA, C. AND MONTENEGRO, G. 2002a. *Ipv6 Opportunistic Encryption*. *INRIA Technical Report Number 4568*.

CASTELLUCCIA, C. AND MONTENEGRO, G. 2002b. Protecting AODVng against impersonation attacks. *ACM Mobile Computing and Communications Review 6*, 3 (July), 2002.

CASTELLUCCIA, C. AND MONTENEGRO, G. 2003. Securing group management in ipv6 with cryptographically generated addresses. In *The Eighth IEEE Symposium on Computers and Communications (ISCC'2003)*.

Crypto-ID JXTA. *Crypto-id jxta* (`http://crypto-id.jxta.org/`).

DEERING, S. AND HINDEN, R. 1998. *Internet Protocol, Version 6 (IPv6) Specification*. IETF, RFC2460.

ELLISON, C. ET AL. 1999. *SPKI Certificate Theory*. IETF, RFC 2693.

HALLER, S. AND METZ, C. 1996. *A One-Time Password System*. IETF, RFC 1938.

HINDEN, B. AND DEERING, S. 2003. *IP Version6 Addressing Architecture*. IETF, RFC3513.

JOHNSON, D., PERKINS, C., AND ARKKO, J. 2003. *Mobile IP for IPv6*. IETF, draft-ietf-mobileip-ipv6-24 (RFC XXX).

JXTA. Project JXTA. Available at `www.jxta.org`.

KAUFMAN, C. 2004. *Internet Key Exchange (IKEv2) Protocol*. IETF, draft-ietf-ipsec-ikev2-12.txt, work in progress.

KIVINEN, T. AND KOJO, M. 2003. *More Modular Exponential (MODP) Diffie-Hellman Groups for Internet Key Exchange (IKE)*. IETF, RFC3526. http://www.join.uni-muenster.de/drafts/draft-nikander-mobileip-v6-ro-sec-00.

KRAWCZYK, H., BELLARE, M., AND CANETTI, R. 1997. *HMAC: Keyed- Hashing for Message Authentication*. IETF, RFC2104.

LENSTRA, A. AND VERHEUL, E. 1999. *Selecting Cryptographic Key Sizes*. Available at `http://citeseer.nj.nec.com/lenstra99selecting.html`.

MADSON, C. AND GLENN, R. 1998. *The Use of HMAC-SHA-1-96 Within ESP and AH*. IETF, RFC2404.

MANKIN, A., PATIL, B., HARKINS, D., NORDMARK, E., NIKANDER, P., ROBERTS, P., AND NARTEN, T. 2001. *Threat Models Introduced by Mobile IPv6 and Requirements for Security in Mobile IPv6*. IETF, draft-ietf-mobileip-mipv6-scrty-reqts-02.txt, work in progress.

MENEZES, A., VAN OORSCHOT, P., AND VANSTONE, S. 1997. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL.

MODADUGU, N., BONEH, D., AND KIM, M. 2000. Generating RSA keys on a handheld using an untrusted server. In *RSA Data Security Conference and Expo, 2000*.

MONTENEGRO, G., LAGANIER, J., AND CASTELLUCCIA, C. 2003. *Cryptographically Generated Addresses (CGA)*. IETF, draft-montenegro-send-cga-rr-01, work in progress.

MOSKOWITZ, B. 2001. *HIP Implementation*. IETF, draft-moskowitz-hip-impl-01.txt, work in progress.

MOSKOWITZ, B. 2003. *HIP Architecture*. IETF, draft-ietf-moskowitz-hip-arch-05.txt, work in progress.

MOSKOWITZ, R., NIKANDER, P., JOKELA, P., AND HENDERSON, T. 2004. *Host Identity Protocol*. IETF, draft-moskowitz-hip-09.txt.

NARTEN, T. AND DRAVES, R. 2001. *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. IETF, RFC3041.

NIKANDER, P. 2001. *An Address Ownership Problem in IPv6*. IETF, draft-nikander-ipng-address-ownership-00.txt, work in progress.

NIKANDER, P., ARKKO, J., AURA, T., AND MONTENEGRO, G. 2003a. Mobile ip version 6 (mipv6) route optimization security design. In *IEEE Vehicular Technology Conference*.

NIKANDER, P., ARKKO, J., AURA, T., MONTENEGRO, G., AND NORDMARK, E. 2003b. *Mobile IP version 6 (MIPv6) Route Optimization Security Design*. IETF, draft-nikander-mobileip-v6-ro-sec-02.txt, work in progress.

NIKANDER, P. AND PERKINS, C. 2002. *Binding Authentication Key Establishment Protocol for Mobile IPv6 (BAKE)*. IETF, draft-perkins-bake-02.txt, work in progress.

NIST 1995. *NIST, FIPS PUB 180-1: Secure Hash Standard*. NIST. Available at `http://www.itl.nist.gov/fipspubs/fip180-1.htm`.

ORMAN, H. 1998. *The OAKLEY Key Determination Protocol*. IETF, RFC2412.

ORMAN, H. AND HOFFMAN, P. 2004. *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*. IETF, draft-orman-public-key-lengths-08.txt, work in progress.

O'SHEA, G. AND ROE, M. 2001. Child-proof Authentication for MIPv6 (CAM). *ACM Computer Communications Review 31*, 2 (Apr.), 4–8.

PERKINS, C., BELDING-ROYER, E., AND DAS, S. 2002. *Ad Hoc On Demand Distance Vector (AODV) Routing for IP version 6*. IETF, draft-perkins-aodv6-02.txt, work in progress.

PERKINS, C., BELDING-ROYER, E., AND DAS, S. 2003. *Ad Hoc On Demand Distance Vector (AODV) Routing*. IETF, RFC 3461.

PERRIG, A. AND SONG, D. 1999. Hash visualization: a new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)*. 131–138.

STAJANO, F. AND ANDERSON, R. 1999. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *7th International Workshop on Security Protocols*. Cambridge, UK.

VAN OORSCHOT, P. AND WIENER, M. 1994. Parallel collision search with applications to hash functions and discrete logarithms. In *Second ACM Conference on Computer and Communications Security*.

Received December 2002; revised November 2003; accepted December 2003

Annexe F

Authentification Secrète

Cette annexe contient un article qui a été publié dans la conférence *ASIACRYPT 2004 : 10th International Conference on the Theory and Application of Cryptology and Information Security, volume 3329. Springer-Verlag GmbH, Decembre 2004.*

Il présente un nouveau protocole qui permet à deux entités de prouver qu'ils appartiennent à la même organisation sans la dévoiler. Ce type de construction peut, par exemple, être utile à des agents des services secrets qui souhaiteraient s'identifier sans révéler leur affiliation à des membres extérieurs à leur organisation.

Secret Handshakes from CA-Oblivious Encryption

Claude Castelluccia, Stanisław Jarecki and Gene Tsudik
School of Information and Computer Science,
UC Irvine, CA 92697, USA
{ccastell,stasio,gts}@ics.uci.edu

Abstract

Secret handshake protocols were recently introduced [1] to allow members of the same group to authenticate each other *secretly*, in the sense that someone who is *not* a group member cannot tell, by engaging some party in the handshake protocol, whether that party is a member of this group. On the other hand, any two parties who *are* members of the same group will recognize each other as members. Thus, a secret handshake protocol can be used in any scenario where group members need to identify each other without revealing their group affiliations to outsiders.

The work of [1] constructed a secret handshake protocol secure under the *Bilinear Diffie-Hellman* (BDH) assumption in the Random Oracle Model (ROM). We show how to build secret handshake protocols secure under more standard cryptographic assumptions, using a novel tool of *CA-oblivious* public key encryption, which is an encryption scheme s.t. neither the public key nor the ciphertext reveal any information about the Certification Authority (CA) which certified the public key. We construct CA-oblivious encryptions and handshake schemes based (in ROM) on either the Computational Diffie-Hellman or the RSA assumption.

keywords: secret handshake, authentication, privacy, anonymity, encryption

1 Introduction

Problem Exposition. A secret handshake scheme, introduced by Balfanz et al. [1], allows two members of the same group to identify each other *secretly*, in the sense that each party reveals his/her affiliation to the other only if the other party is also a group member. For example, a CIA agent Alice might want to authenticate herself to Bob, but only if Bob is also a CIA agent. Moreover, if Bob is *not* a CIA agent, the protocol should not help Bob in determining whether Alice is a CIA agent or not. This secrecy property can be extended to ensure that group members's affiliations are revealed only to members who hold specific *roles* in the group. For example, Alice might want to authenticate herself as a CIA agent with security level one if *and only if* Bob is a CIA agent with security clearance two, and vice versa.

In other words, if A is a member of group G_a with role r_a and B is a member of G_b with role r_b, a secret handshake scheme guarantees the following [1]:

- A and B authenticate each other if and only if $G_a = G_b$.[1]
- If $G_a \neq G_b$ then the only thing that either party learns is *the sole fact* that $G_a \neq G_b$.
- A can choose not to reveal anything about herself unless B is a member with particular role r_b (and vice versa).[2]
- An eavesdropper or a man in the middle learn nothing from the protocol.

[1] However, as noted by [1], a handshake protocol cannot be *fair* in the sense that if $G_a = G_b$ then one party is going to learn about it first and could abort the protocol and thus withold their group affiliation from the counterparty.

[2] To simplify the presentation, we will ignore roles for most of the paper. However, as we show in appendix C.1, they can be easily handled by our schemes.

As observed in [1], secret handshakes seem to require **new cryptographic protocols** since they can not be easily obtained from existing tools in the "cryptographic toolbox". For example,[3] group signatures [2, 3] might appear to be an attractive building block for secret handshakes. However, they offer anonymity and unlinkability of group members' signatures, not secrecy of membership itself. In the interactive variant of group signatures, called *identity escrow* [4], one party can prove to another its membership in a group in an anonymous fashion. However, what turns out to be quite difficult is the seemingly simple issue of two parties proving group membership to each other simultaneously, in such a way that one party never reveals its group membership to another unless the former is also a member of the same group.

Secret Handshake Scheme as a "CA-oblivious PKI". To be usable in practice, a secret handshake scheme must provide efficient revocation of any group member by the Group Authority (GA) which administers the group. To support this functionality we will consider secret handshake schemes which look very much like PKI's (Public Key Infrastructures), where the role of a group authority corresponds to that of a Certification Authority (CA) in a PKI.[4] Namely, to become a member of a group a party needs the GA to issue a certificate on an ID bitstring which the CA agrees to assign to this party. The certificate must include a CA-specific *trapdoor* which corresponds to this ID.[5] To revoke some party, the CA puts that party's ID on a revocation list. To perform a handshake, two parties first exchange their ID's, and then proceed only if the ID of the other party is not on the revocation list of their CA. Since the secret handshake protocol must hide one's group affiliation from outsiders, the IDs will be random strings picked from the same domain by all the CA's.[6]

In this setting, constructing a secret handshake scheme amounts to solving the following protocol problem: For a given CA, Alice wants to prove to Bob that she posseses a trapdoor t_A issued by this CA on her ID_A, but only if Bob posseses a trapdoor t_B issued by *the same* CA on his ID_B (and vice versa). Moreover, the protocol must be "CA-oblivious" in the sense that if a cheating Bob is not in the group administered by a given CA, and hence does not hold a CA-specific trapdoor t_B associated with ID_B, then his interaction with Alice must not help him in guessing if Alice belongs to this group or not. (And vice versa for an honest Bob and a cheating Alice.) While this protocol problem can be solved in principle with general 2-party secure computation techniques, the issue remains whether it can be solved with a *practical* protocol, at a cost comparable to standard authentication protocols.

Existing solution based on bilinear maps. The secret handshake protocol of [1] is based on bilinear maps, which can be constructed using Weil pairings on elliptic curves [5, 6]. The protocol of [1] builds on the non-interactive key-agreement scheme of [7], and works as follows. As in the identity based encryption scheme of [8], A and B can compute each other's public keys from each other's ID's and from the public parameters associated with the CA. If Alice is a group member, she can use her trapdoor t_A corresponding to PK_A to non-interactively compute a session key from (t_A, PK_B). Similarly, if Bob is a group member he can compute *the same* session key from (t_B, PK_A). The two parties can then verify if they computed the same key via a standard MAC-based challenge-response protocol. Under the *Bilinear Diffie-Hellman* (BDH) assumption, it is easy to show (in the Random Oracle Model) that an attacker who does not hold the correct trapdoor cannot compute the session

[3] See [1] for a discussion of the unsuitability of other seemingly related constructs. See also Related Work below.

[4] The SH scheme of [1] also falls in this category.

[5] For example, in an identity based encryption scheme, the trapdoor is a secret key corresponding to the public key which can be recovered from ID and the public parameters associated with the CA. In a standard PKI system, this correspondence has an added level of indirection: The trapdoor t is a secret key corresponding to the public key PK which is in turn bound to the ID string by a signature of CA on the $(ID|PK)$ pair.

[6] Optionally, to make protocol runs executed by the same party *unlinkable*, Balfanz et al. [1] propose that a single ID and its certificate could be used in only one instance of the handshake protocol. The CA would then issue multiple (ID,certificate) pairs to each member.

key. Moroever, the MAC-based challenge response confirmation protocol has the needed property that without the knowledge of the key, one learns nothing from the counterparty's responses.

Thus, the "CA-obliviousness" property of the protocol of [1] follows from two properties of cryptosystems built on bilinear maps: (1) that the receiver's public key can be recovered by the sender from the receiver's ID, and thus the receiver does not need to send any information revealing his CA affiliation to the sender, and (2) knowing their public keys, the two parties can establish a session key non-interactively, and thus they again do not reveal any CA-specific information. Given that the first property relies on identity based encryption, and that the only practical IBE known so far is based on bilinear maps [8],[7] it seems that BDH is indeed needed for secret handshakes.

Our contributions. In this paper we show that efficient secret handshake (SH) schemes can be built using weaker and more standard assumptions than the BDH, namely the *Computational Diffie Hellman* (CDH) and the RSA assumptions, in the so-called Random Oracle Model (ROM).

First, we generalize the IBE-based secret handshake solution described above by showing that an efficient four-rounds secret handshake protocol can be built using any *PKI-enabled* encryption with the additional property of *CA-obliviousness*. We define the notion of (chosen-*plaintext* secure) PKI-enabled encryption, which generalizes both the Identity Based Encryption schemes, and the standard encryption schemes used in the context of a PKI system like X.509. We define the CA-obliviousness property for a PKI-enabled encryption, which says that both the public-key-related information which the receiver provides to the sender, and the ciphertext sent from the sender to the receiver, do not reveal which CA issued the receiver's certificate. We then show that every CA-oblivious PKI-enabled encryption leads to a four-round secret handshake protocol whose cost is one decryption and one encryption for each party.

Next, we combine ElGamal encryption and Schnorr signatures to construct a practical CA-oblivious PKI-enabled encryption secure under the CDH assumption (in ROM), which thus leads to secret handshakes secure under CDH. We also note that the recent "oblivious envelope" construction of [10], which is secure under the RSA assumption (in ROM), can be seen as a secure PKI-enabled encryption, and that with some minor changes it can also be made CA-oblivious and can thus be used to build an RSA-based SH scheme.

Both SH schemes constructed in this manner take four rounds, compared to three rounds in the SH scheme of [1]. However, we show that our CDH-based construction can be simplified to run in three rounds. Moreover, its computational cost is roughly **three to five** times lower than that of the BDH-based SH scheme of [1].

Finally, we show that our SH schemes handle roles just as easily as the SH of [1]. We also show that our CDH-based SH schemes can support "blinded" issuance of the member certificates in the sense that the CA does not learn the trapdoors included in the certificate, and thus, in contrast to the BDH-based SH scheme of [1], the CA cannot impersonate that member.

Related Work. As described in [1], existing anonymity tools such as anonymous credentials, group signatures, matchmaking protocols, or accumulators, have different goals than secret handshakes, and it is indeed unclear how to achieve a secret handshake scheme from any of them. However, the recent work of [10] proposes a notion of "oblivious signature-based envelopes", which is closely related to the secret handshake problem. In fact, the oblivious envelope notion they define is equivalent to a PKI-enabled encryption with *half* of our CA-obliviousness property. Namely, they only require that the identity of the CA is not revealed by the receiver-related information the sender needs to encrypt, while we also require the ciphertext produced by the sender to hide the identity of the CA.

[7] The IBE scheme of [9], whose security rests on the quadratic residuosity decisional assumption, needs one modular group element to encode a single bit of the plaintext. Moreover, we do not know how to establish a shared session key in a CA-oblivious way using this IBE scheme.

The one-sided CA-obliviousness turns out to be enough for the application considered in [10], in which two parties in a regular PKI system announce to each other the cleartexts of their public key certificates, which include the identities of the CA's which signed them, but do not want to reveal to one another the CA's *signatures* on their certificates unless the other party posseses the corresponding signature too. They thus offer a lower level of secrecy protection than a secret handshake would, since, in their protocol a party announces what CA issued its certificate, and they only want to hide the signature which *proves* the possession of this certificate. Nevertheless, the RSA-based encryption scheme given in [10] can be modified to provide the CA-obliviousness in both directions, and thus by our general transformation, it leads to an RSA-based Sescret Handshake scheme.

Organization. In section 2 we revise the definitions of an SH scheme [1], restricting them to "PKI-like" SH schemes we consider here. In section 3 we define the notion of a *PKI-enabled encryption*, and the *CA-obliviousness* property of such encryption. In section 4 we give a general construction of an SH scheme from any CA-oblivious encryption. In section 5 we construct a CA-oblivious encryption secure under CDH in ROM, and we show how to adapt the RSA-based encryption of [10] to make it CA-oblivious. In appendix B we show how to simplify the CDH-based SH scheme so that it is about 3 to 5 times more efficient than the BDH-based SH scheme of [1]. In appendix C we show how to support roles and blinded issuing of CA certificates.

2 Secret Handshake Definition

We adapt the definition of a secure Secret Handshake [SH] scheme from [1] to what we call "PKI-like" SH schemes. Our definitions might potentially restrict the notion of a Handshake Scheme, but both the SH schemes of [1] and ours fall into this category. An SH scheme is a tuple of probabilistic algorithms Setup, CreateGroup, AddMember, and Handshake s.t.

- Setup is an algorithm executed publicly on the high-enough security parameter k, to generate the public parameters params common to all subsequently generated groups.
- CreateGroup is a key generation algorithm executed by a GA, which, on input of params, outputs the group public key G, and the GA's private key t_G.
- AddMember is a protocol executed between a group member and the GA on GA's input t_G and shared inputs: params, G, and the bitstring ID (called a *pseudonym* in [1]) of size regulated by params. The group member's private output is the trapdoor t produced by GA for the above ID.
- Handshake is the authentication protocol, i.e. the SH protocol itself, executed between players A, B on public input ID_A, ID_B, and params. The private input of A is (t_A, G_A) and the private input of B is (t_B, G_B). The output of the protocol for either party is either a $reject$ or $accept$.

We note that AddMember can be executed multiple times for the same group member, resulting in multiple (ID, t) authentication tokens for that member. We also note that in all the SH schemes discussed here the output of the Handshake protocol can be extended to include an authenticated session key along with the "accept" decision.

2.1 Basic Security Properties

A secure SH scheme must be complete, impersonator resistant, and detector resistant:[8]

[8]Once we restrict the notion of SH Schemes to the PKI-like SH schemes, the security properties defined originally in [1] can be stated in a simpler way. Specifically, their properties of *impersonator resistance* and *impersonator tracing* are subsumed by our *impersonator resistance*, and their *detector resistance and tracing* is subsumed by what we call *detector resistance*.

Completeness. If two honest memebers A, B of the same group engage in Handshake with valid trapdoors t_A, t_B generated for their ID strings ID_A, ID_B and for the same group $G_A = G_B$, then both parties output "accept" at the end of the protocol.

Impersonator Resistance. Intuitively, the impersonator resistance property is violated if an honest party V who is a member of group G authenticates an adversary $\mathcal{A}$ as a group member, even though the $\mathcal{A}$ **is not** a member of G. Formally, we say that an SH scheme is *impersonator resistant* if every polynomially bounded adversary $\mathcal{A}$ has negligible probability of winning in the following game, for *any* string ID_V which models the ID string of the victim in the impersonation attack:

1. The Setup and the CreateGroup algorithms are executed: params $\leftarrow$ Setup(1^k), $(G, t_G) \leftarrow$ CreateGroup(params).
2. $\mathcal{A}$, on input (G, ID_V), invokes the AddMember algorithm on any number of group members ID_i of his choice. (The GA's inputs are ID_i's, G, and t_G.)
3. $\mathcal{A}$ announces a new $ID_{\mathcal{A}}$ string, different from all the ID_i's above. (This models a situation where the ID_i's belong to group members who are malicious but who might be revoked.)
4. $\mathcal{A}$ interacts with the honest player V in the Handshake protocol, on common inputs $(ID_{\mathcal{A}}, ID_V)$, and on V's private inputs G and t_V, where $t_V \leftarrow$ AddMember$((G, ID_V), t_G)$.

We say that $\mathcal{A}$ *wins* if V outputs "accept" in the above Handshake instance.

We note that stronger versions of the impersonator resistance property are possible. For example, the attacker can be allowed to ask for trapdoors on additional $ID_i \neq ID_A$ strings *during* the Handshake protocol against the intended victim V. Also, the attacker can be allowed to have several attempts against V and be able to ask for additional trapdoors after each attempt, before he announces that he is ready for the true challenge. We adopt the simpler definition here to reduce the level of formalism, although the schemes we propose remain secure under these stronger notions as well.

Detector Resistance. Intuitively, an adversary $\mathcal{A}$ violates the detector resistance property if it can decide whether some honest party V is a member of some group G, even though $\mathcal{A}$ **is not** a member of G. Formally, we say that an SH scheme is *detector resistant* if there exists a probabilistic polynomial-time algorithm SIM, s.t. any polynomially bounded adversary $\mathcal{A}$ cannot distinguish between the following two games with the probability which is non-negligibly higher than 1/2, for *any* target ID string ID_V:

1-3. Steps 1-3 proceed as in the definition of *Impersonator Resistance*, i.e. on input ID_V and a randomly generated G, $\mathcal{A}$ querries GA on adaptively chosen ID_i's and announces some challenge string $ID_{\mathcal{A}}$, $ID_{\mathcal{A}} \neq ID_i$ for all i.

4-1. In game 1, $\mathcal{A}$ interacts with an algorithm for the honest player V in the Handshake protocol, on common inputs $(ID_{\mathcal{A}}, ID_V)$, and on V's private inputs G and $t_V =$ AddMember$((G, ID_V), t_G)$.

4-2. In game 2, $\mathcal{A}$ interacts with the SIM algorithm on common inputs $(ID_{\mathcal{A}}, ID_V)$. (Note that SIM has no private inputs.)

5. $\mathcal{A}$ can query GA on additional strings $ID_i \neq ID_{\mathcal{A}}$.
6. $\mathcal{A}$ outputs "1" or "2", making a judgment about which game he participated in.

Again, stronger notions of the detector resistance are possible. However, we adopt this relatively simple definition in order to reduce the level of formalism.

2.2 Other Security Properties

Authenticated Key Exchange. In practice one would like to extend the notion of a secret handshake from mere authentication, where participants' outputs are binary decisions "accept"/"reject", to authenticated key exchange, where parties output instead either "reject" or an identifier of an authenticated session and a session key. Our SH schemes, just like the original SH protocol of [1] are in fact easily extendible to AKE protocols. (Formal arguments of AKE security would require extending the AKE formalism of [11, 12], which is beyond the scope of this paper.)

Group-Affiliation Secrecy against Eavesdroppers. Our schemes also protect secrecy of participants' group affiliations against eavesdroppers, even if the eavesdropper is a malicious member of the same group. An observer of our SH protocols does not even learn if the participants belong to the same group or not. We do not formally define security against eavesdroppers, because it is very similar to the security against active attackers which we do define, the impersonator and detector resistance. Moreover, if the protocol participants first establish a secure anonymous session, e.g. using SSL or IKE, and then run the SH protocol over it, the resulting protocol is trivially secure against eavesdroppers.

Unlinkability. A potentially desirable property identified by Balfanz et al. [1], is *unlinkability*, which extends privacy protection for group members by requiring that instances of the handshake protocol performed by the same party cannot be efficiently linked. This can be achieved trivially (but inefficiently) by issuing to each group member a list of one-time certificates, each issued on a randomly chosen ID, to be discarded after a single use. Unfortunately, an honest member's supply of one-time certificates can be depleted by an active attacker who initiates the handshake protocol enough times. Indeed, while one can run our SH schemes using multiple certifciates to offer some heuristic protections against linking, contructing an efficient and perfectly unlinkable SH scheme remains an open problem.

3 PKI-enabled encryption

We define the notion of *PKI-enabled* encryption, which models the use of standard encryption in the context of a PKI system, and also generalizes Identity Based Encryption. We define *one-way security* for PKI-enabled encryption, adapting a standard (although weak) notion of one-way security of encryption to our context, and we define a novel *CA-obliviousness* property for such schemes.

A PKI-enabled encryption is defined by the following algorithms:

- Initialize is run on a high-enough security parameter, k, to generate the public parameters params common to all subsequently generated Certification Authorities (CAs).
- CAInit is a key generation algorithm executed by a CA. It takes as inputs the system parameters params and returns the public key G and the private key t_G of the CA.
- Certify is a protocol executed between a CA and a user who needs to be certified by this CA. It takes CA's private input t_G, and public inputs G (assume that G encodes params) and string ID which identifies the user, and returns *trapdoor* t and *certificate* ω as the user's outputs.
- Recover is an algorithm used by a *sender*, a party who wants to send an encrypted message to a user identified by some string ID, to recover that user's public key. It takes as inputs G, ID, and ω, and outputs a public key PK.
- Encrypt is the actual encryption algorithm which takes inputs message m and the public key PK (assume that PK encodes params and G), and outputs a ciphertext c.

- Decrypt is the decryption algorithm which takes as inputs the ciphertext c and the trapdoor t (as well as possibly params, G, ID, and ω, all of which can be encoded in t), and returns m.

The above algorithms must satisfy the obvious *correctness* property that the decryption procedure always inverts encryption correctly.

It is easy to see (see Appendix A) that this notion of encryption indeed models both regular encryption schemes in the PKI context as well as the Identity Based encryption schemes.

One-Way Security. We define the security of PKI-enabled encryption only in the relatively weak sense of so-called *one-way* security, namely that the attacker who does not own a trapdoor for some public key cannot decrypt an encryption of a random message. This is a weaker notion than the standard *semantic* security for an encryption, but we adopt it here because (1) it simplifies the definition of security, (2) one-way security is all we need in our construction of a secure SH scheme, and (3) in the Random Oracle Model, it is always possible to convert a one-way secure encryption into a semantically secure encryption, or even a CCA-secure encryption using the method of Fujisaki and Okamoto [13].

The definition of security for PKI-enabled encryption is very simliar to the definition of security of an IBE scheme: We say that a PKI-enabled encryption scheme is *One-Way* (OW) secure on message space $\mathcal{M}$ under *Chosen-Plaintext Attack* (CPA), if every polynomially-bounded adversary $\mathcal{A}$ has only negligible probability of winning the following game:

1. The Initialize and CAInit algorithms are run, and the resulting public key G is given to $\mathcal{A}$.
2. $\mathcal{A}$ repeatedly triggers the Certify protocol under the public key G, on ID strings ID_i of $\mathcal{A}$'s choice. In each instance $\mathcal{A}$ receives (t_i, ω_i) from the CA.
3. $\mathcal{A}$ announces a pair $(ID_{\mathcal{A}}, \omega)$, where $ID_{\mathcal{A}} \neq ID_i$ for all ID_i's querried above.
4. $\mathcal{A}$ receives $c = \mathsf{Encrypt}_{PK}(m)$ for a random message $m \in \mathcal{M}$ and $PK = \mathsf{Recover}(G, ID_{\mathcal{A}}, \omega)$.
5. $\mathcal{A}$ is allowed to trigger the Certify algorithm on new $ID_i \neq ID_{\mathcal{A}}$ strings of his choice, getting additional (t_i, ω_i) pairs from the CA.
6. $\mathcal{A}$ outputs a message m'.

We say that $\mathcal{A}$ *wins* in the above game if $m = m'$.

CA-Obliviousness. Informally, PKI-enabled encryption is CA-oblivious if (1) the receiver's message to the sender, i.e., the pair (ID, ω), hides the identity of the CA which certified this ID; and (2) the sender's messages to the receiver, i.e., ciphertexts, do not leak any information about the CA which the *sender* assumed in computing the receiver's public key. Consequentely, in a standard exchange of messages between the receiver and the sender, neither party can guess which CA is assumed by the other one. Formally, we call a PKI-enabled encryption scheme *CA-oblivious* under two conditions:

(I) It is *receiver CA-oblivious*, i.e., if there exists a probabilistic polynomial-time algorithm $SIM_{(R)}$, s.t. no polynomially-bounded adversary $\mathcal{A}$ can distinguish between the following two games with probability non-negligibly higher than 1/2, for *any* target ID string ID_R:

1. The Initialize and CAInit algorithms are executed, and the resulting parameters params and the public key G is given to $\mathcal{A}$.
2. $\mathcal{A}$ can trigger the Certify protocol on any number of ID_i's.

3-1. In game 1, $\mathcal{A}$ gets (ID_R, ω_R), where ω_R is output by the Certify protocol on G and ID_R.

3-2. In game 2, $\mathcal{A}$ gets (ID_R, r) where $r = SIM_{(R)}(\mathsf{params})$.

4. $\mathcal{A}$ can trigger the Certify protocol some more on any $ID_i \neq ID_R$.

5. $\mathcal{A}$ outputs "1" or "2", making a judgment about which game he participated in.

(II) It is *sender CA-oblivious*, i.e., if there exists a probabilistic polynomial-time algorithm $SIM_{(S)}$ s.t. no polynomially-bounded adversary $\mathcal{A}$ can distinguish between the following two games, with probability non-negligibly higher than 1/2:

1. The Initialize and CAInit algorithms are executed, and the resulting parameters params and the public key G is given to $\mathcal{A}$.

2. $\mathcal{A}$ can trigger the Certify protocol any number of times, for public key G and group members ID_i's of $\mathcal{A}$'s choice.

3. $\mathcal{A}$ announces pair (ID_R, ω_R) on which he wants to be tested, where $ID_R \neq ID_i$ for all i.

4-1. In game 1, $\mathcal{A}$ gets $c = \mathsf{Encrypt}_{PK_R}(m)$ for random $m \in \mathcal{M}$ and $PK_R = \mathsf{Recover}(G, ID_R, \omega_R)$.

4-2. In game 2, $\mathcal{A}$ gets $c = SIM_{(S)}(\mathsf{params})$.

5. $\mathcal{A}$ can query GA on some more ID_i's s.t. $\forall_i, ID_i \neq ID_R$.

5. $\mathcal{A}$ outputs "1" or "2", making a judgment about which game he participated in.

4 Secret Handshakes from CA-Oblivious Encryption

We show how to built a secure SH scheme using CA-oblivious PKI-enabled encryption. Given a CA-oblivious one-way secure PKI-enabled encryption scheme (Initialize, CAInit, Certify, Recover, Encrypt, Decrypt), and a hash function $H : \{0,1\}^* \to \{0,1\}^k$ modeled as a random oracle, we specify a secret handshake scheme as follows: Algorithms Setup, CreateGroup, and AddMember, are simply set to Initialize, CAInit, and Certify, respectively, while algorithm Handshake proceeds as follows. A's inputs are (ID_a, ω_a, t_a) and B's inputs are (ID_b, ω_b, t_b).[9]

1. SH-1 $(A \longleftarrow B)$: $ID_b,\ \omega_b$

   A does the following:
   a) obtains $PK_b = \mathsf{Recover}(G, ID_b, \omega_b)$
   b) picks $r_a \leftarrow \mathcal{M}$ and $ch_a \leftarrow \{0,1\}^k$
   c) computes $C_a = \mathsf{Encrypt}_{PK_b}(r_a)$

2. SH-2 $(A \longrightarrow B)$: $ID_a,\ \omega_a,\ C_a,\ ch_a$

   B does the following:
   a) obtains $PK_a = \mathsf{Recover}(G, ID_a, \omega_a)$
   b) obtains $r_a = \mathsf{Decrypt}_{t_b}(C_a)$
   c) picks $r_b \leftarrow \mathcal{M}$ and $ch_b \leftarrow \{0,1\}^k$
   d) computes $C_b = \mathsf{Encrypt}_{PK_a}(r_b)$
   e) computes $resp_b = H(r_a, r_b, ch_a)$

[9] Group member's trapdoor on string ID in this SH scheme is a *pair* (ω, t) produced by the Certify protocol. We can also assume that (ID_a, ID_b) are public inputs.

3. SH-3 ($A \longleftarrow B$): $C_b,\ resp_b,\ ch_b$

   A does the following:

   a) obtains $r_b = \mathsf{Decrypt}_{t_a}(C_b)$

   b) if $resp_b \neq H(r_a, r_b, ch_a)$, outputs FAIL; otherwise outputs ACCEPT.

   c) computes $resp_a = H(r_a, r_b, ch_b)$

4. SH-4 ($A \longrightarrow B$): $resp_a$

   B does the following:

   a) if $resp_a \neq H(r_a, r_b, ch_b)$, outputs FAIL; otherwise outputs ACCEPT.

We note that the above protocol can be easily turned into an Authenticated Key Exchange (AKE) protocol (secure in the ROM model) if the two parties compute their authenticated session key as $K = H(r_a, r_b)$.

Theorem 1 *If the PKI-enabled encryption is CA-oblivious and One-Way (Chosen-Plaintext Attack) Secure, the above construction yields a Secret Handshake scheme secure in the Random Oracle Model (ROM).*

Proof of Impersonator Resistance (sketch): Assume that $\mathcal{A}$ violates with non-negligible probability ϵ the impersonator resistance property against some honest member V identified by ID_V. Assume that $\mathcal{A}$ plays the role of A and V plays the role of B (the other case is easier because B has to speak first). Therefore with prob. ϵ, $\mathcal{A}$ sends a valid $resp_a = H(r_a, r_b, ch_b)$ response to B. In the ROM model, that can happen with non-negligible probability only if $\mathcal{A}$ querries the oracle for $H(\cdot)$ on the input (r_a, r_b, ch_b) s.t., in particular, r_b was the value picked by V and sent to $\mathcal{A}$ in the form of a ciphertext $C_b = \mathsf{Encrypt}_{PK_a}(r_b)$ for $PK_a = \mathsf{Recover}(G, ID_a, \omega_a)$, where (ID_a, ω_a) are sent by $\mathcal{A}$ in its first message to V. Therefore, in ROM, we can use $\mathcal{A}$ to create a break $\mathcal{A}'$ against the one-way security of the encryption scheme:

On input G, $\mathcal{A}'$ passes the public key G to $\mathcal{A}$. When $\mathcal{A}$ can makes a querry ID_i, so does $\mathcal{A}'$, passing back (t_i, ω_i) to $\mathcal{A}$. When $\mathcal{A}$ announces that he is ready for the impersonation challenge against V, $\mathcal{A}'$ passes as *his* encryption challenge the pair (ID_a, ω_a) sent by $\mathcal{A}$ in his first message to V. On encryption challenge $c = \mathsf{Encrypt}_{PK_a}(m)$ where m is chosen at random in $\mathcal{M}$, $\mathcal{A}'$ passes the same challenge as its response $C_b = c$ to $\mathcal{A}$, together with a random challenge value ch_b and $resp_b$ picked at random. The only way $\mathcal{A}$ can tell between this communication and a conversation with an honest V is by querying H on (r_a, r_b, ch_a) for $r_b = \mathsf{Decrypt}_{t_a}(C_b) = m$. Otherwise, as we argued above, he queries H on (r_a, r_b, ch_b) with probability almost ϵ. In either case, since $\mathcal{A}$ can make only polynomially-many queries to H, $\mathcal{A}'$ can pick one such query at random, and $\mathcal{A}'$ will have a non-negligible chance of outputing $r_b = m$. Thus $\mathcal{A}'$ breaks the one-wayness of the encryption scheme. □

Proof of Detector Resistance (sketch): We will show a simulator SIM s.t. if $\mathcal{A}$ distinguishes between interactions with SIM and interactions with a group member, we can break the one-way security of the encryption scheme. Assume again that the adversary $\mathcal{A}$ plays the role of A and V plays the role of B. Assume that the underlying encryption scheme is CA-oblivious, and therefore there exist simulators $SIM_{(S)}$ and $SIM_{(R)}$ which satisfy the two CA-obliviousness criteria. We define a simulator SIM, running on input $(ID_A, ID_V, \mathsf{params})$, as follows: (1) To simulate V's first message SH-1, SIM sends $ID_b = ID_V$ together with $\omega_b = SIM_{(R)}(\mathsf{params})$, (2) To simulate B's second message SH-3, SIM sends $resp_b$ and ch_b picked at random, and $C_b = SIM_{(S)}(\mathsf{params})$.

If $\mathcal{A}$ can distinguish a conversation with such SIM from a conversation with a true group member V, then by a standard hybrid argument, since the $SIM_{(S)}$ and $SIM_{(R)}$ simulators produce messages

which are indistinguishable from the messages of an honest B, it must be that $\mathcal{A}$ distinguishes random values $resp_b$ chosen by SIM from values $resp_b = H(r_a, r_b, ch_a)$ computed by a real player. But this can happen only if $\mathcal{A}$ makes an oracle query on the triple (r_a, r_b, ch_a), in which case we can use $\mathcal{A}$, exactly in the same manner as we did in the proof of impersonator resistance, to attack the one-way security of the underlying encryption scheme. □

5 CA-Oblivious Encryption Schemes: Constructions

We show two constructions for CA-oblivious PKI-enabled encryption schemes, based on the CDH and RSA assumptions, respectively, all in the Random Oracle Model. The CDH-based scheme is novel, while the RSA-based scheme is a simple modification of the RSA-based encryption envelope proposed in [10]. By theorem 1, both constructions lead to secret handshake schemes secure under the respective assumptions.[10]

5.1 CDH-based encryption scheme

- Initialize picks the standard discrete logarithm parameters (p, q, g) of security k, i.e., primes p, q of size polynomial in k, s.t. g is a generator of a subgroup in $\mathbb{Z}_p^*$ of order q. Initialize also defines hash functions $H : \{0,1\}^* \to \mathbb{Z}_q$ and $H' : \{0,1\}^* \to \{0,1\}^k$. (Both hash functions are modeled as random oracles, but we note that H' is not essential in this construction and can be easily removed.)
- CAInit picks random private key $x \in \mathbb{Z}_q$ and public key $y = g^x \mathrm{mod} p$.
- In Certify on public inputs (y, ID), the CA computes the Schnorr signature on string ID under the key y [14], i.e., a pair $(\omega, t) \in (\mathbb{Z}_p^*, \mathbb{Z}_q)$ s.t. $g^t = \omega y^{H(\omega, ID)} \bmod p$. The user's outputs are the trapdoor t and the certificate ω. The signature is computed as $\omega = g^r \bmod p$ and $t = r + xH(\omega, ID) \bmod q$, for random $r \leftarrow \mathbb{Z}_q$.
- Recover(y, ID, ω) outputs $PK = \omega y^{H(\omega, ID)} \bmod p$.
- Encrypt$_{PK}(m)$ is an ElGamal encryption of message $m \in \{0,1\}^k$ under the public key PK: It outputs a ciphertext $[c_1, c_2] = [g^r \bmod p, m \oplus H'(PK^r \bmod p)]$, for random $r \in \mathbb{Z}_q$.
- Decrypt is an ElGamal decryption, outputing $m = c_2 \oplus H'(c_1^t \bmod p)$.

Theorem 2 *The above encryption scheme is CA-oblivious and One-Way (CPA) Secure under the CDH assumption in the Random Oracle Model.*

Proof of One-Way Security (sketch): Assume that an adversary $\mathcal{A}$ breaks one-wayness of this encryption scheme. Then $\mathcal{A}$ receives n Schnorr signatures (t_i, w_i) on ID_i's of his choice. Then $\mathcal{A}$ sends some tuple (ID, w) for $ID \neq ID_i$ for all the above ID_i's. If $\mathcal{A}$ breaks one-wayness then it must be computing $c_1^t \mathrm{mod} p$ where $g^t = \omega y^{H(\omega, ID)} \bmod p$.

Therefore, if $\mathcal{A}$ succeeds, then she can exponentiate a random element c_1 to t. Hence, what we need to argue that, even though $\mathcal{A}$ receives n signatures (t_i, w_i) on her ID_i's, she cannot produce a new pair (ID, w) s.t. she can exponentiate a random elements c_1 to exponent t where $g^t = w * y^{h(t, ID)}$. Now, this is very similar to proving the chosen message attack security of the underlying Schnorr signature scheme, where one argues that, after receiving n signatures, $\mathcal{A}$ cannot produce a new triple

[10] We point out that the Identity Based Encryption scheme of [8] is also a CA-oblivious PKI-based encryption scheme, and therefore our general SH construction applied to that encryption scheme leads to another Weil-pairing based SH scheme.

(ID, t, w) s.t. $g^t = w * y^{h(w,ID)}$. Hence, our proof is very similar to the forking-lemma proof for Schnorr signature security in [15]. However, here we reduce the successful attack not to computing discrete logarithm, but to breaking the CDH assumption by computing m^x on input $y = g^x$ and a random value m.

To reduce $\mathcal{A}$'s ability to succeed in this protocol to computing m^x on the Diffie-Hellman challenge (g, g^x, m), we first simulate, as in the proof of Schnorr signature security, the signatures (t_i, w_i) that $\mathcal{A}$ gets on her ID_i's, by taking random t_i, c_i, computing $\omega_i = g^{t_i} * y^{-c_i} \mathrm{mod} p$, and assigning $H(ID_i, w_i)$ to c_i. Since the verification equation is satisfied and t_i, c_i are picked at random, this is indistinguishable from receiving real signatures. As in the forking lemma argument of [15], we can argue that if $\mathcal{A}$'s probability of success is ϵ, the probability that $\mathcal{A}$ executed twice in a row succeeds in *both* executions *and* sends the *same* (ID, w) pair in both of them, is at least ϵ^2/q_h where q_h is the number of queries $\mathcal{A}$ makes to the hash function H (see [15]). The forking lemma used in the security proof of the Schnorr signature scheme shows that if two conversations with an adversary produce triples (ID, t, w) and $(ID, \hat{t}, w)$, where in first conversation $H(ID, r) = c$ and in the second $H(ID, r) = \hat{c}$ for some random $c, \hat{c}$, then $x = DL_g(y)$ can be computed as $x = (s - \hat{s})/(c - \hat{c}) \bmod q$, because $g^t = w * y^c$ and $g^{\hat{t}} = w * y^{\hat{t}}$. By the same forking lemma applied to our case, adversary $\mathcal{A}$ produces two *exponentiations* m^t and $m^{\hat{t}}$, instead of forgeries $t, \hat{t}$, but still we have that $x = DL_g(y) = (t - \hat{t})/(c - \hat{c})$. Therefore, with probability ϵ^2/q_h we can break the CDH challenge and compute $m^x = m^{(t-\hat{t})/(c-\hat{c})} = (m^t/m^{\hat{t}})^{1/(c-\hat{c})} \bmod p$.

Note that if the success probability ϵ is higher than negligible, and if $\mathcal{A}^*$ is an efficient algorithm and hence the number of queries q_h is polynomial, then the probability of CDH break ϵ^2/q_h is non-negligible as well. □

Proof of CA Obliviousness: It is easy to see that w and the ciphertext $C = [c_1, c_2]$ does not reveal any information about the CA i.e., that the proposed CDH-based encryption is CA-oblivious. Since $\omega = g^r$ for random r, ω is independent from CA's public key y, and hence the scheme is receiver CA-oblivious. Ciphertext $C = [c_1, c_2]$ on a random message m is also independent from the group key y, because $c_1 = g^r$ for random r and c_2 is computed by xoring $H'(PK^r)$ with the random m. □

5.2 RSA-based encryption scheme

- Initialize defines k' polynomial in the security parameter k and two hash functions $H : \{0,1\}^* \rightarrow \{0,1\}^{2k'}$ and $H' : \{0,1\}^* \rightarrow \{0,1\}^k$.
- CAInit picks $e = 3$ and an RSA modulus $n = pq$ for $|p| = |q| = k'$, with additional restriction $2^{2k'} < n < 2^{2k'+1}$. The private RSA key is $d = e^{-1} \bmod \phi(n)$.
- Certify computes the RSA signature on ID under the key (d, n), i.e., $s = H(ID)^d \bmod n$. It then picks $t \leftarrow \mathbb{Z}_n$ and computes $\omega = s * (H(ID))^t \bmod n$. For CA-obliviousness, if $\omega > 2^{2k'}$ then another t is chosen at random and ω is recomputed, until $\omega \leq 2^{2k'}$.
- Recover outputs $PK = \omega^e / H(ID) \bmod n$.
- Encrypt is an ElGamal encryption of k-bit message m under the public key PK modulo n. The sender picks $t \in \mathbb{Z}_n$ and computes the ciphertext $[c_1, c_2] = [(h^e)^r \bmod n, m \oplus H'(PK^r \bmod n)]$, where $h = H(ID)$. Again, for CA obliviousness, if $c_1 > 2^{2k'}$ then another r is chosen at random and ciphertext is recomputed, until $c_1 \leq 2^{2k'}$.
- Decrypt returns $m = c_2 \oplus H'(c_1^t \bmod n)$.

Theorem 3 *The above encryption scheme is CA-oblivious and One-Way (CPA) Secure under the RSA assumption in the Random Oracle Model.*

Proof of One-Way Security: When H' is modelled as a random oracle, then an attack against one-wayness of this encryption scheme means that there exists a polynomially bounded adversary $\mathcal{A}$ s.t. $\mathcal{A}$ gets valid pairs (t_i, w_i) on ID_i's of his choice from the CA, announces a pair (w, ID), s.t. $ID_i \neq ID$, and then the challenger randomly picks $r \leftarrow \mathbb{Z}_n$, and sends $c_1 = (h^e)^r$ and some c_2 to $\mathcal{A}$, where $h = H(ID) \mathrm{mod} n$. The adversary can then receive additional (t_i, w_i) pairs on ID_i's of his choice, but finally $\mathcal{A}$ must make a query $PK^r = (w^e/h)^r \bmod n$ to H, because otherwise, in ROM, he has a negligible chance of outputing the correct decryption of $[c_1, c_2]$.

Given an attacker $\mathcal{A}$ that wins the above game with probability ϵ, we can construct another attacker $\mathcal{B}$ that can successfully forge the RSA signature $H(ID)^d \mathrm{mod} n$ with probability ϵ', where $|\epsilon - \epsilon'|$ is negligible. $\mathcal{B}$ does the following: (1) $\mathcal{B}$ computes $H(ID)$, picks a randon value z in $\mathbb{Z}_n$ and sends $h^{(1+ez)}$ to $\mathcal{A}$. Note that $h^{(1+z)} = h^{(ed+ez)} = h^{e(d+z)}$. $\mathcal{A}$ returns $r = w * e^{(d+z)} * h^{-(d+z)}$. $\mathcal{B}$ can then computes $h^d = w * e^{(d+z)} * h^z / r$.

$\mathcal{B}$ succeeds in forging an RSA signature iff $\mathcal{A}$ wins the above game. This happens with probability ϵ'. What $\mathcal{A}$ receives from the Challenger in the first game is the distributed family $\{h^{er} | r \in \mathbb{Z}_n\}$. What $\mathcal{A}$ receives from $\mathcal{B}$ in the second game is the distributed family $\{h^{e(d+z)} | z \in \mathbb{Z}_n\}$. Since these two distribution families are statistically indistinguishable, $|\epsilon - \epsilon'|$ is negligible. □

Proof of CA Obliviousness: It is easy to see that w and the ciphertext $C = [c_1, c_2]$ does not reveal any information about the CA modulus n. If fact since w is choosen such that its value is always smaller than $2^{2k'}$, the distribution families defined by w for two different values of n, n_1 and n_2, are statistically indistinguishable. The same argument can be used for c_1 and c_2. □

References

[1] D. Balfanz, G. Durfee, N. Shankar, D.K. Smetters, J. Staddon, and H.C. Wong, "Secret handshakes from pairing-based key agreements," in *IEEE Symposium on Security and Privacy*, 2003.

[2] D. Chaum and E. Van Heyst, "Group signatures," in *Advances in Cryptography - EUROCRYPT'91*, Springer-Verlag, Ed., 1991, vol. 547, pp. 257–265.

[3] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," in *CRYPTO'2000*, 2000.

[4] J. Kilian and E. Petrank, "Identity escrow," in *Advances in Cryptography - CRYPT0 1998*, Santa Barbara, CA, August 1998.

[5] A. Joux, "The weil and tate pairings as building blocks for public key cryptosystems," in *Proceedings of the 5th International Symposium on Algorithmic Number Theory*, 2002.

[6] Martin Gagne, "Applications of bilinear maps in cryptography," M.S. thesis, University of Waterloo, 2002.

[7] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *Symposium in Cryptography and Information Security*, Okinawa, Japan, January 2000.

[8] D. Boneh and M. Franklin, "Identity based encryption from weil pairing," in *Advances in Cryptography - CRYPT0 2001*, Santa Barbara, CA, August 2001.

[9] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Eighth IMA International Conference on Cryptography and Coding*, Decembre 2001.

[10] N. Li, W. Du, and D. Boneh, "Oblivious signature-based enevelope," in *Proceedings of 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, Boston, Massachusetts, July 13-16 2003.

[11] R. Canetti and H. Krawczyk, "Universally composable notions of key exchange and secure channels," in *Advances in Cryptology - EUROCRYPT 2002*, 2002.

[12] Shoup V., "On formal models for secure key exchange," Tech. Rep. RZ3120, IBM, April 1999.

[13] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Advances in Cryptology-CRYPTO'99*, August 1999, pp. 537–554.

[14] C. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptography - CRYPT0 1989*, Santa Barbara, CA, August 1989.

[15] D. Pointcheval and J. Stern, "Security proofs for signatures," in *Eurocrypt'96*, 1996, pp. 387 – 398.

[16] P. Barreto, H. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Advances in Cryptography - CRYPT0 2002*, Santa Barbara, CA, August 2002.

A PKI-enabled encryption generalizes both standard PKI and IBE

The notion of the PKI-enabled encryption generalizes both the standard use of encryption in the PKI context and the Identity Based Encryption schemes. An IBE scheme can be seen as a special case of PKI-enabled encryption where certificates ω computed in the Certify protocol are empty strings. Therefore, a given user's public key PK is computable solely from G (CA's public key) and ID (the user's identity).

Similarly, if we want to model the standard PKI setting, the Initialize procedure would simply output as params its input 1^k. This output sets the *minimal* security parameter to be used by any CA participating in this PKI system. The CAInit generates the standard public/private signature keypair (CA, t_{CA}) of the CA. The Certify procedure generates a private/public keypair (Pr, PK) for the user, and outputs the user's trapdoor $t = Pr$ and a certificate $\omega = (PK, sig_{CA}[PK, ID])$, which binds the user's ID to his public key. Then Recover(CA, ID, ω) outputs PK if the signature in ω verifies under public key CA on message $[PK, ID]$. Otherwise Recover outputs null. Encryption and decryption proceed in a standard way.

B More Efficient CDH-based Secret Handshake

This section presents a secure extension of the *CDH-based* SH scheme presented above that uses "CA-oblivious Diffie-Hellman Key Agreement" instead of "CA-oblivious Diffie-Hellman encryption".

This extension improves efficiency so that the scheme has only 3 rounds (instead of 4) and requires only one (multi)exponentiation per player. This results in a SH scheme which has the same number of rounds but requires at least 3 times computationnally more efficient than the *BDH-based* SH scheme of [1]. In fact as shown in [16], for the same level of security, a Weil pairing computation is about 5 to 6 times more expensive than a modular exponentation computation, i.e. at least 3 times more expensive than a modular multi-exponentation.

The faster secret handshake protocol is as follows:

1. Initialize picks the standard discrete logarithm parameters (p, q, g) of security k, i.e., primes p, q of size polynomial in k, s.t. g is a generator of a subgroup in $\mathbb{Z}_p^*$ prime order q. Initialize also defines hash functions $H : \{0,1\}^* \to \mathbb{Z}_q$ and $H' : \{0,1\}^* \to \{0,1\}^k$.

2. CAInit picks random private key $x \in \mathbb{Z}_q$ and public key $y = g^x \bmod p$.

3. In Certify on public inputs (y, ID), the CA computes the Schnorr signature of ID under the key y [14], i.e., the signature on ID is a pair $(\omega, t) \in (\mathbb{Z}_p^*, \mathbb{Z}_q)$ s.t. $g^t = \omega * y^{H(\omega, ID)} \bmod p$. The user's outputs are the trapdoor t and the certificate ω.

4. Recover(y, ID, ω) outputs $PK = \omega * y^{H(\omega, ID)} \bmod p$. The message space $\mathcal{M} = \{0,1\}^k$.

5. The Handshake protocol:

   The secret handshake protocol between Alice on inputs t_A, (w_A, ID_A), and y, and Bob on inputs t_B, (w_B, ID_B), and y, proceeds as follows:

   (a) msg1 $(A \longrightarrow B)$: w_A, ID_A, N_1

   B computes $M_A = w_A y^{h(w_A, ID_A)}$ (from Recover) and $K_B = M_A^{t_B} \bmod p$, computes $v_B = mac(K_B, \tau)$ where $\tau = (w_B || ID_B || N_2 || msg1)$

   (b) msg2 $(B \longrightarrow A)$: w_B, ID_B, N_2, v_B

   $\mathcal{A}$ computes $M_B = w_B y^{h(w_B, ID_B)}$ and $K_A = M_B^{t_A} \bmod p$, $v_A = mac(K_A, \tau')$ where $\tau' = (msg1 || msg2)$, and accepts the authentication protocol if $v_B = mac(K_A, \tau)$.

   (c) msg3 $(A \longrightarrow B)$: v_A

   B accepts the authentication protocol if $v_A = mac(K_B, \tau')$

We show that the above scheme achieves the *detector resistance and impersonator resistance* properties under the CDH assumption in the Random Oracle Model, and therefore we have:

Theorem 4 *In the Random Oracle Model, under the CDH assumption, the above secret handshake protocol is* secure.

Intuitively, we show that to remain *secure* a malicious party $\mathcal{A}$ would have to use a $w_{\mathcal{A}}$ which was not issued by the CA. We argue that in this case, the ability of $\mathcal{A}$ to authenticate itself (in the Random Oracle Model) is equivalent to his ability to compute the Diffie-Hellman key $K = M_B^{t_A^*}$ where t_A is defined by $g^{t_A^*} = \mathsf{Recover}(w_A^*, y)$ and $M_B = \mathsf{Recover}(w_B, y)$. We then show that his ability to compute such exponentiation is equivalent to solving a computational Diffie-Hellman challenge and computing z^x on input g, y, z, where $y = g^x$.

Interestingly, the proof of the secrecy property is very similar to the proof of security. The ability of an adversary $\mathcal{A}$, who does not have a valid group member certificate under the group key y, to tell if its participant in the secret handshake protocol is a member of the group under this public key, is equivalent to $\mathcal{A}$ being able himself to compute the proper authentication tag $tag = H(K, \tau)$, after sending some value w_A^* which is again not issued by the CA. But in the Random Oracle Model this is again equivalent to $\mathcal{A}$'s ability to compute the Diffie-Hellman key K as above. Hence the same proof as the one in the Appendix for theorem 4 shows that existance of such adversary is equivalent to breaking the computational Diffie Hellman problem.

Proof Assume that an adversary $\mathcal{A}$ receives n authentication tokens (t_i, w_i) from a group of malicious colluding group members $\{A_i\}$. This is equivalent to $\mathcal{A}$ receving n modified-Schnorr (message,signature) pairs $(ID_i, (t_i, w_i))$. Suppose that $\mathcal{A}$ successfully authenticates in the secret handshake protocol to some honest party B. If $\mathcal{A}$ sends one of the received legimitate certificates (w_i, ID_i) to B in the authentication protocol, the CA will immediately identify this certificate. Suppose then that $\mathcal{A}$ sends some (w, ID) different from any of the received certificates. We show that in that case, under the CDH assumption, the probability that $\mathcal{A}$ the authentication protocol is negligible. The reason is that if $\mathcal{A}$ passes the authentication protocol then he must compute the right authentication tag value $v = h(K, \tau)$ where $K = M^{t * t'} \bmod p$, where t' is the Diffie-Hellman secret held by B, and

$M = g^t \mathrm{mod} p$ is defined as $\mathsf{Recover}((w, ID), y) = wy^{h(w,ID)} \mathrm{mod} p$. In the Random Oracle Model of analysis, computing such value is possible, except for negligible guessing probability, only by querying the hash function on the correctly computed argument K. Therefore, in the ROM model, an adversary $\mathcal{A}$ who succeeds in this protocol must also output $K = M^{t'} = g^{t*t'}$ as a query to the hash function. Note that $\mathcal{A}$ receives information about t' in the form of B's certificate $w', ID')$ which defines t' because it defines $M' = \mathsf{Recover}((w', ID'), y)$ where $M' = g^{t'}$. In other words, $\mathcal{A}$ commits himself to t by sending $M = g^t$ (in the form of values (w, ID)), receives a random M' (in the form of values (w', ID')), and outputs $K = M'^t$.

Therefore, if $\mathcal{A}$ succeeds then he can exponentiate a random element M' to value t committed to in M. Hence, what we need to argue in the proof of traceability is that even though $\mathcal{A}$ receives n (message,signature) pairs $(m_i, (w_i, t_i))$ he cannot produce a new element (t, m) s.t. he can exponentiate random elements M' to exponent t where $g^t = M = wy^{h(w,m)}$. Now, this is very similar to proving the chosen message attack security of the underlying modified-Schnorr signature scheme, where one argues that, after receiving n (message,signature) pairs, $\mathcal{A}$ cannot produce a new triple (t, w, m) s.t. $g^t = wy^{h(w,m)}$. The proof is very similar to the forking-lemma proof for Schnorr signature security in [15], but here we reduce the successfull attack not to computing discrete logarithm, but to breaking the CDH assumption by computing M'^x on input $y = g^x$ and a random value M'. (We will later on with the fact that our $\mathcal{A}$ does not get M' as such, but in the form of w', m'. For now assume that $\mathcal{A}$ receives from B a random value M' in the authentication protocol.)

To reduce $\mathcal{A}$'s ability to succeed in this protocol to computing M'^x on the Diffie-Hellman challenge (g, y, M'), we first simulate, as in the proof of Schnorr signature security, the signatures (t_i, w_i, m_i) that $\mathcal{A}$ gets by taking random t_i, m_i, c_i, computing $w_i = g^{t_i} * y^{-c_i} \mathrm{mod} p$, and assigning $h(w_i, m_i)$ to c_i. Since the verification equation is satisfied and c_i is picked at random, this is indistinguishable from receiving random (message,signature) pairs. As in the forking lemma argument of [15], we can argue that if $\mathcal{A}$'s probability of success is ϵ, the probability that $\mathcal{A}$ executed twice in a row succeed in *both* executions *and* he sends the *same* $C = (w, m)$ certificate in both of them, is at least ϵ^2/q_h where q_h is the number of queries $\mathcal{A}$ makes to the hash function h (see [15] for the argument). The signature-scheme security argument shows that if with probability at least ϵ^2/q_h, two conversations with $\mathcal{A}$ produce triples (m, t, w) and $(m, \hat{t}, w)$, where in first conversation $h(m, w) = c$ and in the second $h(m, w) = \hat{c}$ for some random $c, \hat{c}$, then $x = DL_g(y)$ can be computed as $x = (t - \hat{s})/(c - \hat{c})$ because $g^s = wy^c$ and $g^{\hat{t}} = wy^{\hat{c}}$, and therefore $y = g^{(t-\hat{t})/(c-\hat{c})}$. Here, however, by the same forking lemma $\mathcal{A}$ does not produce two forgeries t and $\hat{t}$, but two *exponentiations* $K = M'^t$ and $\hat{K} = M'^{\hat{t}}$, but still we have that $x = DL_g(y) = (t - \hat{t})/(c - \hat{c})$ because here again wea have that $M = g^t = wy^c$ and $\hat{M} = g^{\hat{t}} = wy^{\hat{c}}$. Therefore, with probability ϵ^2/q_h we break the CDH challenge and compute M'^x as $(K/\hat{K})^{1/(c-\hat{c})} = (M'^{t-\hat{t}})^{1/(c-\hat{c})}$.

Note that if the success probability ϵ is higher than negligible, and if $\mathcal{A}$ is an efficient algorithm and hence the number of queries q_h is polynomial, then the probability of CDH break ϵ^2/q_h is non-negligible as well.

Finally, we have to deal with the fact that $\mathcal{A}$ does not compute $K = M'^t$ on any random M' he is given, but on $M' = w'y^{h(m',w')}$ where (m', w') is sent to $\mathcal{A}$ by B. Therefore the reduction from CDH is slightly different but still succesful: For example, we can still break a CDH input challenge g, y, w' and compute w'^x by running the above simulation twice, using the same values m', w' in the two simulations, but different values for $h(m', w')$: c' and $\hat{c'}$. From the first succesful simulation we get $M'^x = w'^x y^{xc'}$ (because a conversation with $\mathcal{A}$ can be used to recover $\alpha = M'^x$ in the same way as argued above), and from the second we get $\hat{\alpha} = \hat{M}'^x = w'^x y^{x\hat{c'}}$. Therefore $\alpha^{\hat{c'}}/\hat{\alpha}^{c'} = w'^{x\hat{c'}}/w'^{xc'} = (w'^x)^{(\hat{c'}-c')}$, and hence we can break CDH and output w'^x by exponentiating the above entity to $1/(\hat{c'} - c')$. □

C Achieving Additional Security Properties

This section presents a few simple enhancements of the above SH schemes. We show that our SH schemes handle roles just as easily as the SH of [1]. We show that our CDH-based SH schemes can support "blinded" issuance of the member certificates in the sense that the CA does not learn the trapdoors included in the certificate, and thus the CA cannot impersonate that member.

C.1 Roles

Our schemes can easily be extended to handle group member roles (as in the SH scheme of [1]), in a way that a member can choose not to reveal anything about herself unless the other party is a member with a particular role r (and vice versa). This functionality can be provided by modifiying the AddMember and Recover procedures as follows:

- AddMember: takes as inputs $params$, G, t_G and an arbritary string $ID \in \{0,1\}^*$ and returns (t, ω) where t is a trapdoor and ω is a public parameter. (t, ω) are constructed using the string $ID|r$ (instead of ID as in the original procedure), where r is the role that the CA is assigning to the user.

- Recover: takes as input $params$, G, ID and ω (provided by another user B). It outputs a public key PK using as input $ID|r$ (instead of ID as in the original Recover procedure), where r is the role that $\mathcal{A}$ chooses to have a secret hanshake with.

C.2 Trapdoor Secrecy

Note that in the RSA-based solution, the trapdoor can be computed by the user as part of AddMember and does not have to be known to the CA.

In the CDH-based solution, however, the trapdoor t is computed by the CA. As a result, the CA can impersonate the user or eavesdrop on his communications. Would that be problematic, AddMember can easily be modified to blind the trapdoor as follows:
AddMember: takes as inputs $params$, G, t_G (only known to the CA), $b = g^{\delta}$, where δ is a secret only known by the user, and an arbitrary string $ID \in \{0,1\}^*$. The CA computes $w = g^k * b$, where k is a random value in Z_{p-1}, and $t' = k + H(ID|w) * t_G$. The user then computes his trapdoor $t = t' + \delta$ (note that (t, ω) is a valid ElGamal signature of ID).